UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY


UNIVERSIDAD DE LA REPÚBLICA

PROYECTO DE GRADO

---

# Qode - Herramienta para el análisis cualitativo de datos

---

*Autores:*
Carina Soca
Santiago Camou
Nicolás Urruty

*Supervisor:*
Ing. Regina Motz

*Informe de Proyecto de Grado presentado al Tribunal Evaluador como requisito de graduación de la carrera Ingeniería en Computación*

Instituto de Computación

3 de mayo de 2018

***A nuestros compañeros.***
*Con los que hemos compartido incontables horas de clase y estudio, además de litros y litros de mate acompañado de bizcochos de queso. Por compartir tanto logros como frustraciones.*
***A nuestros amigos.***
*Que tantas veces han tenido que entender que no podíamos estar. Los que crecieron con nosotros y los que encontramos en este camino.*
***A nuestras familias.***
*Los que nos acompañan todos los días desde siempre. Especialmente nuestros padres, por todo lo que trabajaron por nuestra educación y calidad de vida. Por mostrarnos la ruta, pero dejarnos crear nuestro camino.*

# *Agradecimientos*

En primer lugar queremos agradecer a nuestra tutora, Ing. Regina Motz, por el tiempo brindado. Por facilitarnos la información y contactos necesarios para avanzar en nuestro proyecto. Gracias por sus correcciones y por el respeto a nuestras sugerencias e ideas.

Luego agradecer a la Dra. Cristina Gerascoff que desde su lugar de cliente nos dio su *feedback* siempre que lo necesitamos y, aún más, buscó personas externas al proyecto para que probaran el sistema.

Nuestro sincero agradecimiento a Alén Pérez Casas y Luis Pablo Alonzo, sin ellos el entendimiento del área y herramientas de análisis cualitativo hubiese sido muy complejo. Con su conocimiento y experiencia nos ayudaron a comprender de qué se trataba una herramienta de estas características, además de apoyar nuestro proyecto desde un principio.

Gracias a, Joe Blankenship, desarrollador y *data scientist*, quien nos brindó su conocimiento para ayudar a diseñar un sistema adecuado al problema.

A todos ellos, muchas gracias.

«*Cuando la información se organiza, surgen las ideas.*»

Jim Rohn

UNIVERSIDAD DE LA REPÚBLICA

# *Resumen*

Facultad de Ingeniería
Instituto de Computación

Ingeniería en Computación

**Qode - Herramienta para el análisis cualitativo de datos**

por
Carina Soca
Santiago Camou
Nicolás Urruty

El análisis cualitativo de datos es una fase que se enmarca dentro del proceso de investigación cualitativa. En este informe se presenta un prototipo de una herramienta de *software* para asistir en el análisis cualitativo de datos. Esta herramienta, que la hemos dado en llamar "Qode", está desarrollada en *software* libre como un servicio *web* y presenta funcionalidades para anotar códigos en documentos de texto, permitiendo además el trabajo colaborativo entre distintos investigadores. Qode está pensado para escalar a volúmenes grandes de texto poco estructurados y se implementó utilizando la base de datos MongoDB. Las funcionalidades desarrollados atienden a la cliente del proyecto, la Dra. en Odontología bioenergética María Cristina Gerascoff quien realiza una investigación cualitativa en talleres para personas que desean combatir el bruxismo. Independientemente de esta aplicación en el área de salud, la herramienta Qode aplica a cualquier área de estudio dónde se necesite realizar el marcado de códigos en textos. Adicionalmente la herramienta Qode presenta también funcionalidades de recuperación y visualización de los códigos utilizados y sus ocurrencias. En este informe se presenta el análisis de las funcionalidades de una herramienta de este tipo y las decisiones tomadas durante la implementación de la herramienta que se desarrolló.

**Palabras clave:** Investigación cualitativa, Análisis cualitativo, CAQDAS. Herramienta para Análisis Cualitativo de Datos.

# Índice general

# Índice de figuras

# Índice de cuadros

# Lista de Abreviaciones

| | |
|---|---|
| **API** | **A**pplication **P**rogramming **I**nterface |
| **CAQDAS** | **C**omputer-**A**ided **Q**ualitative **D**ata **A**nalysis **S**oftware |
| **CPU** | **C**entral **P**rocessing **U**nit |
| **CRUD** | **C**reate **R**ead **U**pdate **D**elete |
| **HTML** | **H**yper**T**ext **M**arkup **L**anguage |
| **JSON** | **J**ava**S**cript **O**bject **N**otation |
| **ML** | **M**achine **L**earning |
| **MVC** | **M**odel **V**iew **C**ontroller |
| **NoSQL** | **N**o **S**tructured **Q**uery **L**anguage |
| **NLP** | **N**atural **L**anguage **P**rocessing |
| **RAM** | **R**andom **A**ccess **M**emory |
| **UI** | **U**ser **I**nterface |
| **UX** | **U**ser e**X**perience |
| **XML** | e**X**tensible **M**arkup **L**anguage |

# Capítulo 1

# Introducción

El análisis cualitativo de datos es una actividad fundamental en la realización de investigaciones en diversas ramas de la ciencia, principalmente las ciencias sociales, la cuales abordan las distintas problemáticas y desarrollan teorías en base a estudios cualitativos por sobre cuantitativos.

El *software* existente hoy en día le proporciona al investigador diversas funcionalidades que facilitan el trabajo manual que se realiza convencionalmente en estas actividades. Sin embargo, la mayoría de las herramientas utilizadas actualmente son *software* propietario con alto costo para la investigación. Por otro lado, desde el desarrollo informático tradicional se atiende principalmente a la investigación cuantitativa con el desarrollo de diferentes tipos de herramientas y por tratarse de una ciencia de las llamadas "duras" el análisis cualitativo de datos no está aún incoporado en los planes de estudio. A través del trabajo de investigación cualitativa sobre la relación del bruxismo con el estrés y sus consecuencias de la odontóloga Cristina Gerascoff surge el interés y la posibilidad de estudiar y desarrollar una herramienta para asistir en el análisis cualitativo de datos.

Este trabajo se enfoca primeramente en comprender la esencia del análisis cualitativo de datos y el trabajo de la Dra. Gerascoff, para luego comparar el *software* existente en el mercado y a partir de estos obtener las características básicas que debe tener una herramienta de esta índole. Como resultado de este trabajo se presenta un primer prototipo de una herramienta para asistir al análisis cualitativo de datos desarrollada con *software* libre, que hemos dado en llamar Qode (juego de las palabras: "Qualitative-code").

A continuación introducimos el concepto de la investigación cualitativa, su metodología y el aporte de las herramientas informáticas a esta área. Presentamos el trabajo que realiza la Dra. Gerascoff en relación al bruxismo, y como puede beneficiarse de la herramienta propuesta.

## 1.1. ¿Qué es análisis cualitativo de datos?

Si bien muchas veces al mencionar análisis cualitativo de datos, se interpreta como toda la investigación cualitativa, cabe hacer la diferenciación entre ésta última como el proceso, que comprende desde la recolección de datos hasta la obtención de resultados y el análisis como la etapa en la que exclusivamente se analizan los datos ya obtenidos y almacenados.

La investigación cualitativa se llama así porque se realiza sobre datos cualitativos. Según Hernández Pina, García-Sanz y Maquilón éstos por definición son datos no cuantitativos, es decir, que no puede ser expresado como números. Por lo tanto, son difícilmente medibles, no traducibles a términos matemáticos y no sujetos a la inferencia estadística.

"Taylor y Bogdan (1986) sostienen que la investigación cualitativa es aquella que produce datos descriptivos: las propias palabras de las personas, habladas o escritas, y la conducta observable"(citado por Herrera, 2008, p. 7). Según Rodríguez, Gil y García (1996, p. 32) esta rama de la investigación estudia la realidad en su contexto natural donde ocurren los fenómenos estudiados tal y como suceden. Se intenta interpretar los mismos y el significado que tiene para las personas implicadas. El formato típico de estos datos son textos abiertos (por lo general entrevistas, historias de vida, textos históricos) e imágenes, y pueden estar registrados de todas las formas posibles siendo las más comunes archivos de texto, audio o video.

La investigación cualitativa surge inicialmente de las ciencias sociales, siendo la sociología y la antropología sus principales ramas impulsoras. Sin embargo hoy en día trasciende estas áreas siendo extensamente aplicada en medicina y psicología, y potencialmente aplicable a innumerables temáticas de investigación. "Según Lincoln y Denzim (1994:576), la investigación cualitativa es un campo interdisciplinar, transdisciplinar y en muchas ocasiones contradisciplinar. Atraviesa las humanidades, las ciencias sociales y las físicas. La investigación cualitativa es muchas cosas al mismo tiempo. Es multiparadigmática en su enfoque." (citado por Herrera, 2008, p. 7)

Por las propias características de los datos en cuestión, la investigación cualitativa es muy distinta de la cuantitativa. La diferencia más fácil de reconocer es que los datos cualitativos no son valores exactos, por lo tanto su significado dista mucho de ser universal y absoluto, por el contrario, el valor de dichos datos es relativo al contexto en que se obtienen y a la interpretación que el investigador hace sobre los mismos. Sin embargo, la diferencia más importante entre ambos enfoques no viene dada por el tipo de dato sino por donde se sitúa el punto de interés del investigador.

> "Para Stake (1995), la primera característica diferenciadora de la investigación cualitativa no se asienta en el enfrentamiento entre dato cualitativo versus dato cuantitativo, sino que se sitúa el terreno epistemológico. El objetivo de la investigación cuantitativa es la comprensión, centrando la indagación en los hechos; mientras que la investigación cualitativa fundamentará su búsqueda en las causas, persiguiendo el control y la explicación. Desde la investigación cualitativa se pretende la comprensión de las complejas interrelaciones que se dan en la realidad."(citado por Rodriguez et al.1996, p. 12)

En otras palabras, la cualitativa investiga el por qué y el cómo se tomó una decisión, en contraste con la investigación cuantitativa, que busca responder preguntas más deterministas cómo dónde y cuándo ocurrió y cuál fue el resultado. Su propósito es explorar las relaciones sociales y describir la realidad tal como la experimentan los protagonistas.

## 1.2. Metodología del análisis cualitativo de datos

Los métodos cualitativos no han sido tan estandarizados como otros enfoques investigativos: "Miles y Huberman ( 1994: 5-8), consideran como características básicas de la investigación naturalista las siguientes: (...) Se utilizan relativamente pocos instrumentos estandarizados. El investigador es el principal instrumento de medida."(citado por Rodriguez et al.1996, p. 11). Se siguen lineamientos orientadores, pero no reglas. Según Schettini y Cortazzo (2015) los investigadores deberían "sentirse

libres" para inventar métodos y estrategias que le permitan develar las problemáticas que plantean investigar. Los métodos sirven al investigador, pero éste es libre de adaptarlos a su preferencia de forma que le sean más útiles, aún así sin perder validez.

> "Cualquiera sea el método que usemos para analizar siempre será una interpretación de lo analizado. Al analizar aislamos porciones de discurso, hacemos una selección particular de la información que posiblemente otro investigador no haría, es más, inferimos lo no dicho; esta selección de frases se apoya no solo en nuestra experiencia de vida, nuestro mundo sociocultural sino también en nuestra intuición y, fundamentalmente, en nuestros objetivos e hipótesis" (Schettini y Cortazzo, 2015, p. 86).

De todas formas el investigador tiene que evitar influir en la obtención de datos, de modo que estos sean lo más fieles posible a la realidad observada. Por ejemplo, en caso de una entrevista el investigador debe evitar influir en el entrevistado en su forma de preguntar y debe evitar "pre-analizar" los datos ya que esto puede producir un sesgo para el posterior análisis, no teniendo un panorama completamente objetivo de lo que el entrevistado expresó. "Para Strauss Corbin (1998), el investigador cualitativo debe ser 'conocedor de los datos y la teoría, y al mismo tiempo capaz de escapar los aspectos de su propio trabajo que puedan bloquear la nueva perspectiva, el presentimiento, la intuición, la idea brillante, o la formulación teórica diferente' " (citado por Alarcón Venegas, Carrasco Quijada y Pérez Ponce, 2012)

Como se observa en la Figura1.1, la investigación cualitativa se compone de varias etapas. Este proyecto se focaliza en la fase análitica del proceso de investigación cualitativa: se trabaja con datos preexistentes, concentrándose en desarrollar una herramienta que colabore con el análisis y representación de los resultados. Sin embargo, como destacan Rodríguez et al.(1996, p. 64), los lìmites entre las distintas fase son difusos, las etapas se suceden unas a otras, pero este proceso no es lineal. Como se observa en la representación gráfica, existen superposiciones, hay procesos de ida y vuelta entre las distintas fases, ocurriendo solapamientos y procesos de retroalimentación de la información producida en cada fase. En las siguientes subsecciones se realiza una breve descripción de etapas de la investigación cualitativa, siguiendo a Rodríguez et al.(1996).

1. **Fase preparatoria**
   Lo primero que debe hacer el investigador es diseñar el estudio a realizar, determinar los objetivos de la investigación y la metodología más adecuada para alcanzarlos. La investigación se plantea como un proceso de relacionar conceptos teóricos existentes con otros descubiertos a lo largo de la misma. A partir de esas relaciones se genera un modelo sobre el proceso y el contenido del cambio.

2. **Trabajo de campo**
   Se entiende como un proceso en el que el investigador va accediendo progresivamente a la información fundamental para su estudio. La recogida de la información hay que realizarla de acuerdo a los requerimientos de la estrategia de investigación.

3. **Fase analítica**
   "Definimos el análisis de datos como un conjunto de manipulaciones, transformaciones, operaciones, reflexiones, comprobaciones que realizamos sobre

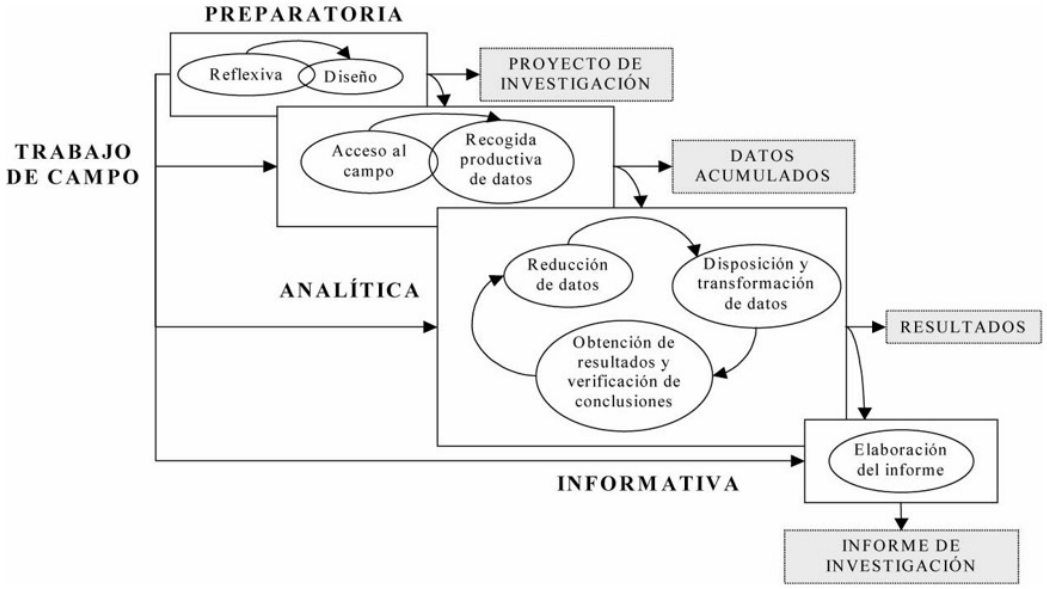



FIGURA 1.1: Fases investigación cualitativa(tomada de Rodriguez et al.,1996).

lo datos con el fin de extraer significado relevante en relación a un problema de investigación" (Rodriguez et al.1996, p. 200).

Aunque esta fase es posterior al trabajo de campo, la necesidad de contar con una investigación con datos suficientes y adecuados exige que las tareas de análisis comiencen durante el mismo.

Resulta difícil hablar de una estrategia o procedimiento general de análisis de datos cualitativos ya que varía bastante según el investigador.

> "Según Tesch (1990) citado por Rodríguez Gómez y otros (1999: 202): 'Hasta tal punto se diversifica la forma de realizar el análisis que se ha llegado a afirmar que el único punto de acuerdo entre los investigadores es la idea de que el análisis es el proceso de extraer sentido de los datos'" (citado por Schettini y Cortazzo, 2015, p. 16).

Sin embargo, hay una serie de tareas que constituyen el proceso básico en la mayoría de los estudios cualitativos. Las mismas según Miles y Huberman, 1994 serían: a) reducción de datos; b) disposición y transformación de datos y c) obtención de resultados y verificación de conclusiones.

La metodología más usada es la de identificación de conceptos en los datos (codificación o categorización), para luego establecer relaciones entre los conceptos encontrados. Esto con el objetivo de construir una teoría en torno a dichos conceptos.

4. **Fase Informativa**
   En esta etapa el investigador prepara la comunicación de los resultados obtenidos. Según Rodríguez et. al.(1996, p. 76) el proceso de investigación culmina con la presentación y difusión de los resultados. El objetivo de esta etapa es compartir con distintos auditorios la sistematización de la información obtenida y someterlos a discusión.

## 1.3. Análisis cualitativo de datos asistido por computadora

CAQDAS es el acrónimo de Computer-Aided Qualitative Data Analysis, que en español podría traducirse como Software de Análisis Cualitativo de Datos Asistido por Computadora. Los CAQDAS son programas computacionales que le permiten a los investigadores cualitativos, de cualquier disciplina, ordenar los documentos que deseen analizar, de la forma que consideren más adecuada. Van desde simples editores de texto con búsqueda de palabras hasta herramientas específicas muy completas, que facilitan tareas como realizar anotaciones, segmentar, categorizar y codificar los datos, efectuar análisis sobre la ocurrencia de palabras o categorías interrelacionadas en los datos, visualizar del proceso de análisis por parte de terceros, exportar resultados obtenidos, etc.

Los programas informáticos para la ayuda del análisis cualitativo son herramientas que permiten un manejo mucho más fácil de los datos, aumentan la productividad del investigador permitiendo realizar investigaciones sobre un volumen de datos mucho más grande y en tiempos razonables. Según Alén Pérez[1] (entrevista personal, 6 de Junio, 2017), se considera que el potencial impacto de la informática para los métodos cualitativos es aún mayor que el alcanzado para los cuantitativos, ya que en general los datos en el primer caso son mucho más extensos y por lo tanto menos manejables de forma manual. Además, ante la preocupación de algunos sectores de investigadores que creían que estos métodos quitarían validez a las investigaciones, se ha visto que por el contrario las investigaciones aumentan su validez ya que muchas de estas herramientas mantienen un registro de las acciones realizadas. Esto permite ver paso a paso como fue realizado el análisis y de donde surgen los resultados, habilitando la trazabilidad del proceso analítico.

Las anteriores no son las únicas ventajas que ofrecen los CAQDAS, también se incluyen capacidades que una herramienta informática le suma a cualquier proceso. Algunas de ellas son: trabajo colaborativo (tanto de forma sincrónica como asincrónica), vínculos con información en la web, modelado y visualización de resultados, posibilidad de importar datos de otros proyectos (incluyendo el trabajo realizado sobre los mismos), algunos proyectos han evaluado la posibilidad de que el programa ayude directamente en la tarea analítica mediante sugerencias o incluso procesos automatizados. Este último punto es muy discutido y bastante resistido ya que se cree que el proceso de análisis cualitativo es muy subjetivo para ser realizado mediante inteligencia artificial, para lo cual todo tipo de ambigüedad en la información puede generar resultados no deseados o teorías que no son válidas. Lee y Fielding (1998) clasifican a los *software* para el análisis de datos cualitativos en tres tipos básicos, los mismos son:

1. **Los recuperadores de texto**
   Permiten realizar búsquedas de ocurrencias de palabras, frases y combinaciones de éstas que se quieran localizar en uno o más archivos del proyecto. Muchos de ellos también permiten realizar análisis del contenido, pero sobre todo cuantitativo como conteo de ocurrencias y frecuencia de aparición de palabras.

2. **Los codificadores y recuperadores**
   Contienen en su gran mayoría las funcionalidades mencionadas en la categoría anterior, pero a estas le agregan funcionalidades propias de la investigación

[1] Magister en Sociología de la Facultad de Ciencias Sociales, Universidad de la República. Especializado en Informática Aplicada a la Enseñanza y a la Investigación Social. Investigador de los Impactos Sociales de las Tecnologías de la Información y la Comunicación. Participante de proyecto LibreQDA

cualitativa. Permiten segmentar y categorizar los datos asignándoles códigos. Facilitan las tareas de recuperación, agrupación y ordenación de segmentos y códigos.

3. **Constructores de teoría**
   Estos programas surgen en el marco de la Grounded Theory (Teoría Fundamentada o Enraizada) de Glaser y Strauss (1967). Su principal diferencia con los codificadores, es que estos generan relaciones entre los códigos además de generar relaciones con los datos. Lógicamente incluyen todas las funcionalidades de las categorías anteriores a las que agregan otras nuevas que facilitan la construcción teórica, es decir, que permiten, a través del análisis y codificación de los datos, encontrar relaciones entre los códigos e inferir conocimiento de las mismas. Estos programas están organizados alrededor de un sistema de reglas o basados en la lógica formal, y ofrecen búsquedas booleanas o posibilidades de probar hipótesis.

La herramienta Qode que desarrollamos en este proyecto entra en la categoría anterior, codificadores y recuperadores.

## 1.4. La cliente y su trabajo

Este proyecto surge por interés de la Dra. Mª Cristina Gerascoff (odontóloga) para utilizarlo en experiencias de investigación, de corte grupal y comunitario. A continuación se describe brevemente en qué consiste el trabajo de la cliente y como una herramienta de análisis cualitativo le puede ser de utilidad.

Dra. Mª Cristina Gerascoff Azambuya, doctora en odontología bioenergética. Actualmente docente del Master de Terapia neural médica y odontológica, de la Universidad de Barcelona, autora del libro Sanar desde la Boca, dictante de cursos y talleres en Uruguay, España, Argentina, Colombia y Puerto Rico. Pionera en odontología bioenergética en Uruguay trabaja en su consultorio privado en Atlántida, Canelones.

Participación como docente en el curso del Manejo del estrés asociado a TTM[2], desde el año 2007 al 2013 inclusive, en la escuela de postgrado de la Facultad de Odontología de la UDELAR. Dictado de talleres vivenciales a distintos grupos de salud comunitaria en relación al tratamiento domiciliario del bruxismo.

### 1.4.1. Breve descripción del Bruxismo

El bruxismo lo padece un alto porcentaje de la población en Uruguay y está asociado al estrés y a la dificultad de hacer un manejo apropiado del mismo. El bruxismo trae frecuentemente muchas consecuencias que pueden afectar gravemente a la salud de las personas. A nivel dentario puede producir, desgaste exagerado y prematuro de los dientes, problemas cervicales, fracturas dentarias con pérdida de los dientes afectados, problemas articulares en las articulaciones témporomandibulares, neuralgias a nivel facial, cefaleas, contracturas en el cuello y espalda, dolores y adormecimiento de brazos entre otros. Todos estos síntomas frecuentemente no son vinculados a su trastorno de origen por quienes los padecen, y en muchas ocasiones tampoco por los profesionales de la salud. Esto hace que se atiendan los síntomas sin alivio duradero, llegando a situaciones verdaderamente penosas en la salud de la gente afectada.

[2]Trastorno temporomandibular

### 1.4.2. Talleres sobre el Bruxismo

La cliente realiza talleres comunitarios sobre técnicas de manejo del estrés y reprogramación neuromuscular para el tratamiento del bruxismo. Estas son experiencias piloto, que promueven políticas de salud comunitaria en el área de la odontología bioenergética en el municipio de Atlántida.

Por ejemplo: Taller de musicoterapia donde se usa el canto y la expresión como herramienta para facilitar el manejo de esta dificultad aportando al auto-tratamiento individual y comunitario con dirección y supervisión profesional.

En estas instancias se usan grupos focales, entrevistas individuales y colectivas, encuestas en papel para obtener datos sobre las sensaciones y síntomas de los participantes del taller a través de las anotaciones y resultados de encuestas obtenidos. En esta recolección de datos, subjetiva y participativa, se busca obtener una evaluación sobre la metodología del taller en sí mismo, la receptividad de los participantes, las dificultades en su desarrollo, y a la vez la influencia de los talleres en el área específica del bruxismo, el manejo del estrés y otros aspectos que surjan del material obtenido. La Dra. luego realiza un análisis cualitativo sobre estos datos, identificando conceptos y buscando relaciones entre los mismos que se repiten en varios casos. Aquí es donde Qode resulta de utilidad, facilitando el manejo de múltiples archivos de texto a la vez y ayudando en el proceso de codificación e identificación de relaciones, logrando una vinculación multifacética entre las distintas metodologías de obtención de datos (entrevistas, encuestas, trabajos grupales, bitácoras de los talleres, etc.) y logrando una observación conjunta y a la vez específica, además de una representación gráfica que permite visualizar y compartir el análisis de datos.

Los requerimientos que plantea la cliente, descritos concretamente en el Capítulo 3, son particulares, ya que ella no tiene experiencia en el uso de CAQDAS. El proceso de análisis que realiza no está enmarcado en una metodología formalizada, es un trabajo más desestructurado y que se ajusta al tipo de análisis que se aplica en los programas más usados en el área, la codificación de segmentos de texto. Por tales motivos surge la idea de darle a Qode una mirada más general del problema. La necesidad particular planteada por la Dra. no presenta restricciones que sean específicas a la odontología o a la salud. Una herramienta de análisis cualitativo de propósito general se ajusta a las necesidades planteadas y a su vez puede ser de gran utilidad para futuros usuarios.

## 1.5. Objetivos del proyecto Qode

El objetivo inicial de este proyecto es crear una herramienta que sea de utilidad para la cliente en su tarea de realizar el análisis cualitativo a partir de datos obtenidos en distintos talleres sobre salud relacionados al área de la odontología.

Sin embargo, como se vio en el Capítulo 1, la investigación cualitativa, y más particularmente, el análisis cualitativo son multidisciplinarios y multi-paradigmáticos. Esto fundamenta la idea de darle a la herramienta, un enfoque más general y no tan específico al área en cuestión, para que la misma solución sea de utilidad para otras personas, pero aún satisfaciendo las necesidades del cliente.

Otro punto a destacar, viendo el panorama actual sobre uso de programas para asistir la investigación cualitativa, es que hay una gran variedad. Como se verá en la Capítulo 2 algunas de estas herramientas son muy completas y ofrecen gran cantidad de funcionalidades. El tamaño del equipo y los plazos manejados, vuelve inviable plantearse como un objetivo realista el desarrollo de un *software* capaz de competir con los antes presentados.

Ante esta imposibilidad, es que surge el segundo objetivo. Este es, aprovechando las características comunitarias del *software* libre, lograr desarrollar un programa que sea una base firme para que en un futuro, con aportes de la comunidad, se logre una herramienta tan completa como las actuales líderes en el área. Existen varios casos de *software* libre que impulsados por la comunidad llegaron a posicionarse entre las herramientas líderes en sus respectivas áreas. Algunos son Linux[3], LibreOffice[4] , R[5] o Mozzilla[6], por nombrar los más conocidos.

Que el programa sea *software* libre no va en que se brinde como servicio gratuitamente. « *'Free software'* es el *software* que respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el *software*. Es decir, el '*software* libre' es una cuestión de libertad, no de precio.» (*¿Qué es el software libre?*, GNU). Como medio para completar el objetivo en cuestión se requiere que Qode cumpla con las libertades que debe cumplir un *software* para ser considerado libre. En el Apéndice G se profundiza sobre las características del *software* libre.

Aún así, lograr que este proyecto genere buen arraigo, capte el interés de actores de esta comunidad y llame la atención en investigadores cualitativos por su potencial, no es una tarea fácil. Se debe implementar una solución escalable, que permita fácil incremento de funcionalidades, que también funcione en los principales sistemas operativos (para ampliar el espectro de potenciales usuarios) y en lo posible funcione de la misma forma en todos ellos, sin necesidad de versiones distintas para cada caso. Además, para lograr la aceptación de los usuarios, el sistema debe ser muy intuitivo, de fácil uso y que no sea una herramienta sólo para expertos en análisis cualitativo; que sea una opción para todos los que se interesen por un CAQDAS, desde principiantes a expertos. Lograr una herramienta de fácil aprendizaje, podría utilizarse con fines didácticos en usuarios que están adentrándose a la investigación cualitativa. De esta forma, se puede generar una comunidad de buena cantidad de usuarios que pueda incentivar futuras extensiones de la herramienta, siguiendo los ideales de "*software* libre" los cuales pregonan que los usuarios deben controlar el programa y su evolución.

### 1.5.1. Resumen de objetivos

De lo expuesto anteriormente se extrae el siguiente listado de objetivos:

1. Crear una herramienta que satisfaga las necesidades del cliente, facilitando el análisis de la investigación en relación a su campo de estudio.

2. Desarrollar una aplicación libre y escalable que pueda utilizarse como base por parte de la comunidad para crear un CAQDAS competitivo con las herramientas existentes en el mercado.

3. Lograr un programa de uso intuitivo y fácil aprendizaje.

Como resultado de estos objetivos se implementó la herramienta Qode. El código fuente junto con su manual de instalación se puede encontrar en **nurruty.github.io/qode/**. En **qodeapp.herokuapp.com** esta disponible un prototipo funcional del sistema.

[3] getgnulinux.org
[4] es.libreoffice.org/
[5] www.r-project.org
[6] www.mozilla.org/es-ES/

El resto de este documento se organiza de la siguiente forma.

El Capítulo 2 realiza un relevamiento de las herramientas existentes en el mercado más utilizadas actualmente y se presenta una comparación de las mismas marcando sus ventajas y desventajas relativas.

El Capítulo 3 se enfoca en determinar el alcance del proyecto desde un punto de vista del desarrollo de *software*, primeramente analizando las funcionalidades básicas de una herramienta de este tipo y luego exponiendo aquellas que se implementaron.

El Capítulo 4, presenta la descripción técnica de la herramienta desarrollada. Se analizan las decisiones técnicas que debieron tomarse en cada instancia.

Finalmente, en el Capítulo 5 se concluye el trabajo, presentando las ideas finales del equipo y planteando posibles extensiones y mejoras que se le pueden realizar al sistema.

# Capítulo 2

# Relevamiento de herramientas

Este capítulo presenta el relevamiento realizado sobre las funcionalidades que ofrecen las herramientas para análisis cualitativo de datos más utilizadas hoy en día.

En la actualidad, existen diversas herramientas para análisis cualitativo de datos en el mercado, la mayoría de ellas software propietario y a pesar de existir algunas propuestas de *software* libre y código abierto éstas últimas no han sido ampliamente adoptadas hasta el momento. El conjunto de herramientas que seleccionamos para analizar surgió a partir de las recomendaciones de instituciones educativas que usan este tipo de programas, tales como las universidades de Boston[1] , Illinois[2], Nuevo México [3], Surrey [4] y Wisconsin[5] . Esto coincide con la evaluación que hacen los investigadores y docentes de la Universidad de la República, los cuales en gran mayoría optan por el uso de Atlas.ti. Si bien es imposible lograr un consenso general sobre cuál es la mejor, ya que depende de la metodología, propósito y preferencia general del usuario, estas son algunas que suelen aparecer entre las más destacadas.

Por otro lado, existen algunos desarrollos de herramientas open source, entre ellos AQUAD [6] , CAT(Coding Analysis Toolkit)[7], Compendium [8], ELAN [9] y la propuesta del LibreQDA. Nos interesa especialmente analizar la propuesta LibreQDA por tratarse de una iniciativa que surgió en un proyecto donde participaban docentes de la Facultad de Psicología de la Universidad de la República, la empresa Tryolabs [10] y la Unviversidad Autónoma de Barcelona, proyecto que ha quedado congelado por más de 3 años. Al final de esta sección se evalúa esta herramienta observando el estado actual del proyecto, los objetivos propuestos inicialmente por sus desarrolladores y posibilidades de avance a futuro.

## 2.1. Principales herramientas

### 2.1.1. QDAMiner

QDAMiner[11] es una herramienta muy completa para el análisis cualitativo. Acepta una muy amplia gama de tipos de archivo, ofrece muchas funcionalidades y tiene

[1] bu.edu/tech/services/cccs/desktop/distribution/nvivo/comparison
[2] guides.library.illinois.edu/c.php?g=348074&p=2346110
[3] methodologygroup.unm.edu/methods/caqdas.html
[4] surrey.ac.uk/computer-assisted-qualitative-data-analysis/resources
[5] web.education.wisc.edu/qdatools/guides/choosing-software
[6] aquad.de/es/
[7] cat.texifter.com/
[8] compendiuminstitute.net/
[9] tla.mpi.nl/tools/tla-tools/elan/
[10] https://tryolabs.com/
[11] provalisresearch.com

la capacidad de generar distintos tipos de visualización de resultados según las necesidades del usuario.

Sin embargo, el problema de QDAMiner es que es un *software* pago. Si bien hay una versión liviana gratuita, esta ofrece muy pocas funcionalidades. De hecho, si se compara la versión gratuita con la paga, la primera ofrece solamente una de las ocho herramientas de análisis y recuperación de codificación que ofrece la segunda.

**Características clave**

- **Codificación en pantalla y anotaciones de documentos**: esta herramienta ofrece buena flexibilidad y facilidad de uso. Permite realizar acciones como la división de código, fusión, fácil cambio de tamaño de segmentos codificados, búsqueda y reemplazo interactivos de código o agrupación virtual y flexible escritura de memos. Esto sirve para asociar piezas de evidencia cualitativa mediante la creación de vínculos a otros segmentos codificados, casos, documentos, archivos o sitios *web*.
- **Herramientas de geoetiquetado y de etiquetado de tiempo**: Permite asociar coordenadas temporales y geográficas a segmentos de texto o áreas gráficas, recuperar datos codificados con base en el tiempo o ubicación. También representar eventos en el espacio, crear mapas dinámicos y líneas de tiempo interactivas.
- **Codificación en pantalla y anotaciones de documentos**: esta herramienta ofrece buena flexibilidad y facilidad de uso. Permite realizar acciones como la división de código, fusión, fácil cambio de tamaño de segmentos codificados, búsqueda y reemplazo interactivos de código o agrupación virtual y flexible escritura de memos. Esto sirve para asociar piezas de evidencia cualitativa mediante la creación de vínculos a otros segmentos codificados, casos, documentos, archivos o sitios *web*.
- **Herramientas de búsqueda de texto**: búsqueda por palabra clave, recuperación de sección y una herramienta de extracción de conglomerado y de codificación. Estas herramientas son de gran utilidad a la hora de codificar los documentos, ya que se puede buscar palabras de interés muy rápidamente en todos los documentos del proyecto. Estas funcionalidades son esenciales teniendo en cuenta que los proyectos pueden contener una enorme cantidad de documentos, explorarlos y/o realizar búsquedas exhaustivas sería muy difícil sin ellas.
- **Herramientas de recuperación de codificación**: estas sirven para extraer segmentos codificados relacionados con códigos específicos o patrones de código e identificación de co-ocurrencias de codificación, secuencias de codificación y evaluación de las relaciones entre la codificación y propiedades numéricas o categoriales.
- **Herramientas estadísticas y de visualización**: tales como clasificación, escalamiento multidimensional, mapas térmicos, análisis de correspondencia y de secuencia que permiten identificar rápidamente patrones y tendencias, explorar datos, describir, comparar y probar hipótesis. Esto es clave a la hora de poder representar los resultados obtenidos de forma amigable y entendible.
- **Capacidad de trabajo multiusuario**: una función de fusión para traer codificación, anotaciones, informes y entradas en el registro de varios codificadores, así

como un módulo de acuerdo entre codificadores para evaluar la confiabilidad de la codificación. Posee un registro de comandos para realizar el seguimiento o auditoría de cada acceso al proyecto, operación de codificación, transformación, consulta y análisis realizado. Esto sirve para documentar el proceso de análisis cualitativo y para supervisar el trabajo en equipo.

- **Administración de reportes**: permite almacenar en una sola ubicación las consultas y análisis de resultados, tablas, gráficos, notas y citas. Sirve para organizar los resultados e interpretaciones, ayudando a los investigadores cualitativos en el proceso de escritura del reporte.

#### Restricciones

QDAMiner presenta un problema y es que solamente está disponible para sistemas operativos Windows. En caso de tener que usarlo en un sistema Linux o MacOS es necesario instalar una máquina virtual para ejecutar el programa, lo cual no es un impedimento total pero sí un aspecto a considerar.

### 2.1.2. Atlas.ti

ATLAS.ti[12] es un potente *software* para el análisis cualitativo, posee un conjunto de herramientas para grandes cuerpos de datos textuales, gráficos y de vídeo. Según sus creadores: " La sofisticación de las herramientas le ayuda a organizar, reagrupar y gestionar su material de manera creativa y, al mismo tiempo, sistemática. ATLAS.ti le permite mantenerse centrado en el propio material de investigación." Dentro de su rama, es uno de los programas de mayor difusión a nivel global, aunque al igual que la mayoría de ellos no es gratuito y adquirir una licencia puede resultar bastante costoso.

#### Características clave

- **Múltiples formatos:** Atlas.ti permite trabajar con la mayor parte de los formatos de archivos de texto incluyendo txt, doc, docx, odt, pdf, etc. Pero no sólo permite trabajar con texto, ya que también soporta archivos multimedia para los cuáles permite la carga en decenas de formatos. En comparación con herramientas similares Atlas es una de las que tiene mayor amplitud en el universo de formatos aceptados.

- **Contenido semántico:** Un diferencial que posee esta herramienta es el poder darle un significado semántico a los códigos, como por ejemplo poder especificar si dos códigos son opuestos. Esto trae consigo un gran potencial a la hora de construir una teoría basada en el análisis realizado.

- **Búsquedas:** Atlas.ti ofrece al igual que casi todas las herramientas similares, una buena variedad de búsquedas rápidas de ocurrencia de palabras o de códigos dentro de todos los documentos del proyecto. Estas búsquedas pueden ser además de co-ocurrencias, lo que permite indagar sobre el nivel de relacionamiento entre dos códigos. Además otra herramienta presente, la cual no es tan común, es el *auto coding*, esto implica poder realizar búsquedas de ocurrencias de patrones sintácticos en los documentos y codificarlas automáticamente con un código especificado por el usuario.

[12] atlasti.com/es/

- **Interoperabilidad:** La cantidad de formatos en los que se puede importar y exportar los datos de un proyecto en este programa hacen que se pueda integrar con una gran variedad de herramientas semejantes. HTML Y XML son dos formatos libres y estandarizados en los que el usuario puede exportar los datos. Pudiendo generar representaciones en HTML, el usuario tiene gran facilidad en caso de querer publicar los resultados en la *web*. Además Atlas no sólo permite la integración de sus datos a otras herramientas generando archivos XML, sino que ha sido uno de los principales impulsores de la creación de estándares de estructuración de datos en XML para herramientas de este tipo. Esta herramienta permite también generar resultados en formatos como Excel y permite la carga de documentos enriquecidos con hipervínculos a la *web* u objetos incrustados como tablas de Excel o diapositivas de Powerpoint.

- **Trabajo en equipo:** Atlas.ti ofrece la posibilidad de participación multi-usuario en los proyectos,donde varios analistas pueden trabajar de forma simultánea con distintos perfiles y permisos. Esto se traduce en un aumento de transparencia, debido al fortalecimiento de la comunicación online del equipo de investigación.

- **Visualización:** Atlas.ti permite una amplia gama de formas de visualización de resultados, tanto gráficos representativos de resultados cuantitativos, como frecuencias de aparición de palabras o códigos, como también de ocurrencias de palabras, nubes de palabras, pero el diferencial más interesante es la red de relaciones de objetos. Esto es un grafo que muestra las relaciones entre todo tipo de objetos dentro del proyecto, códigos, palabras, documentos, anotaciones, etc, todo aparece en la misma red conformando un verdadero mapa del proyecto que resulta muy útil a la hora de construir la teoría.

- **Facilidad de uso:** Una característica muy importante de este tipo de herramientas es qué tan fácil resulta para un usuario usar la interfaz, ya que por más que se presenten funcionalidades muy completas y con gran potencial, esto no sirve de nada si usarlas se vuelve una labor muy tediosa o difícil de aprender. Es por esto que Atlas da muchas facilidades a la hora del manejo del espacio de trabajo, presentando gran cantidad de funcionalidades en los márgenes de la pantalla, las cuales son en gran medida realizando un sólo click o arrastrando un elemento de su posición. El espacio de trabajo es además fácilmente adaptable al gusto de investigador, ya que éste puede dividir la pantalla en diferentes regiones permitiendo ver más de un documento en simultánea (función no muy habitual en herramientas similares), e incluso tener abierta la lista de códigos, anotaciones, documentos,etc.

**Restricciones**

Algo que puede ser visto como una restricción dentro de Atlas.ti es que no es posible relacionar los códigos de forma jerárquica, los códigos están todos al mismo nivel. Este tipo de funcionalidad es ofrecida en varias otras herramientas, y no parece ser muy difícil de implementar en comparación con otras que sí se ofrecen en este programa, lo que hace pensar que esto se debe al concepto que se tiene en esta herramienta sobre la forma de realizar el análisis.

### 2.1.3. MAXQDA

"*Software* profesional para la investigación cualitativa y métodos mixtos" es la frase de presentación en el sitio oficial de MaxQDA[13].

**Características Clave**

- **Múltiples fuentes de datos:** MAXQDA permite importar y analizar entrevistas, datos de grupos focales, de encuestas por internet o de páginas *web*. También le permite importar archivos de imágenes, de audio y de vídeo, hojas de cálculo, datos bibliográficos y hasta *tweets* de Twitter [14]. Dentro de estas funcionalidades una que destaca por no ser muy común en el resto de los programas de análisis cualitativo, es la herramienta de transcripción. La misma permite regular la velocidad de reproducción o el volumen de los archivos multimedia incluso con pedales USB, de modo de optimizar el análisis y la transcripción de los datos.

- **Orientación a métodos mixtos:** Los creadores de este SOFTWARE lo proclaman como "el especialista en métodos mixtos". Para respaldar tal atribución ofrecen algunas funcionalidades como la posibilidad de establecer vínculos entre datos cualitativos y variables sociodemográficas. De manera similar, se pueden cuantificar los resultados de un análisis cualitativo o calcular frecuencias estadísticas de manera sencilla y directa. Además de estas funcionalidades, existe una versión extendida del *software* llamada MAXQDAplus que incluye el complemento MAXDictio, el cual añade la funcionalidad de análisis de contenido cuantitativo en el análisis de textos. De este modo es posible analizar más fácilmente el vocabulario o contenido de textos.

- **Visualización de resultados:** Esta herramienta da la posibilidad de generar visualizaciones de datos de manera automática o manual, proporcionando distintas formas que van desde matrices de co-ocurrencias de códigos hasta gráficas y nubes de palabras. Los resultados generados pueden ser compartidos a través de la exportación de archivos de imagen que ya salen preparados para su uso en informes y presentaciones. Además, es posible crear mapas conceptuales con la herramienta integrada MAXMapas.

- **Interfaz intuitiva:** MaxQDA mantiene desde hace años su estructura de interfaz dividida en 4 pantallas cada una de ellas correspondiente a alguna de las áreas de trabajo esenciales en el proceso del análisis cualitativo de datos:

  - El conjunto completo de sus datos (ventana: "grupo de textos")
  - El sistema de códigos o categorías (ventana: "sistema de códigos")
  - El texto simple, accesible siempre y directamente (ventana: "visualizador de texto") y eventualmente
  - Realización de búsquedas básicas y complejas, al verificar sus conceptos teóricos (ventana: "segmentos recuperados")

  Además de esto, la usabilidad se ve muy facilitada por el acceso sencillo a muchas de las funcionalidades con solo un click, o arrastrando elementos en la pantalla.

[13] www.maxqda.com/
[14] twitter.com

### 2.1.4. NVivo

NVivo[15] es uno de los *software* de análisis cualitativo más usados. Según su sitio oficial: "NVivo es un *software* que se dirige a la investigación con métodos cualitativos y mixtos. Está diseñado para ayudarlo a organizar, analizar y encontrar perspectivas en datos no estructurados o cualitativos, como: entrevistas, respuestas de encuestas con preguntas abiertas, artículos, contenido de las redes sociales y la *web*."

**Características clave**

- **Gran variedad de formatos de documento:** Esta herramienta permite importar y analizar documentos de varios tipos como PDF, hojas de cálculo, bases de datos, audio, video o imágenes. Y para cada tipo de documento tolera una buena variedad de formatos. Un aspecto importante a destacar es que NVivo tiene limitado el espacio que se puede utilizar para los recursos de cada proyecto, sin embargo al importar archivos de medios, estos se pueden guardar dentro o fuera del proyecto, de forma de aprovechar el espacio disponible.
- **Administrar datos bibliográficos:** Permite integración con aplicaciones bibliográficas como EndNote [16] o Zotero [17], se puede importar los datos de la aplicación a NVivo. Los registros se agregan como recursos(documentos) en el proyecto y los detalles bibliográficos se almacenan como clasificaciones de recursos y valores de atributo.
- **Análisis de datos estructurados:** Por datos estructurados se entiende conjuntos de recursos que presentan una misma estructura fija. Por ejemplo, encuestas con las mismas preguntas para un conjuntos de individuos. NVivo permite visualizar los datos de forma correspondiente agrupando las respuestas a cada una de las preguntas.
- **Generación de redes:** Permite la generación de redes, son gráficos que se crean a partir de vínculos que se establecen entre diferentes elementos, algunas de ellas son: mapas mentales, mapas conceptuales y mapa de proyecto.
- **Agrupamientos:** Esta herramienta da la posibilidad de organizar en carpetas los elementos del proyecto (recursos, memos, nodos, entre otros), las cuáles son llamadas familias. La interfaz brinda la facilidad de poder organizar los elementos simplemente arrastrándolos a la carpeta destino.
- **NCAPTURE:** Este complemento permite importar al programa recursos de distintos sitios *web*, puede ser desde un sitio entero a determinado archivo multimedia (como un video de Youtube[18] o una imagen), como también de redes sociales se puede importar publicaciones puntuales. Funciona como una extensión en el *browser* facilitando la extracción al realizarse directamente mientras se navega.
- **Visualizaciones:** El programa permite visualizar los datos analizados mediante el uso de gráficas para presentar los datos en el proyecto y también diagramas de exploración, comparación, conglomerado y de mapa ramificado.

[15] qsrinternational.com/nvivo/home
[16] endnote.com
[17] zotero.org
[18] youtube.com

- **Realización de transcripción:** tiene su propio programa incorporado para hacer la transcripción de los datos procedentes de formatos de audio y video. Puede reproducir y pausar los medios, transcribir a medida que escucha, mientras el programa crea automáticamente nuevas filas y marcas de tiempo. Como ayuda, permite la utilización de un pedal de pie y accesos directos del teclado, además al momento de codificar el material transcripto, se puede hacer directamente en los medios (mediante la línea de tiempo) o codificar la transcripción. NVivo ofrece también un servicio de transcripción, el mismo es pago pero ofrece una integración completa con la herramienta, obteniendo el resultado una vez realizada la misma sin necesidad de salir del programa.

- **Búsquedas:** La herramienta de búsqueda de NVivo posee la opción de buscar un fragmento de texto o una sola palabra, así como también combinaciones lógicas de ocurrencias de palabras, bien sea en todos los recursos o solo en uno. Además existe la posibilidad de codificar automáticamente los resultados de una búsqueda con un código seleccionado.

- **Compartir proyectos en NVivo Server:** NVivo Server posibilita el trabajo colaborativo entre distintos investigadores en el mismo proyecto al mismo tiempo. Con esta aplicación, miembros del equipo de trabajo que tengan distintas versiones del *software* (Starter, Pro,etc) pueden trabajar sobre un mismo proyecto. Es una aplicación no gratuita.

- **Presentación de resultados:** este programa ofrece gran variedad de opciones para la realización de informes y la presentación de resultados directamente, sin pasar por otros programas. Mediante los llamados extractos se permite exportar un grupo de datos a un texto, archivo Excel o XML. Además la herramienta posee un diseñador de informes para ayudar al usuario en la elaboración de sus conclusiones e informes de avance.

- **Codificación automática:** NVivo posee varias funciones de codificación automática. Está la posibilidad de codificación por temas, donde el programa se encarga mediante identificación de frases que se repiten en los recursos hacer cierta clasificación de temas, a partir de los cuales crea códigos (denominados nodos). Otra posibilidad es la de análisis de sentimiento, en este caso el proceso utiliza un sistema de puntuación, donde cada palabra que contiene un sentimiento posee una puntuación predefinida. El contenido de los recursos se codifica en un conjunto de nodos de sentimientos, que varían desde muy positivo a muy negativo.

- **Análisis de redes sociales:** No solo se permite importar publicaciones de las redes sociales como recursos, también se puede usar todo tipo de datos de redes sociales, permitiendo generar redes de relaciones entre individuos estudiados como casos, viendo que relaciones y menciones en publicaciones hay entre ellos, pudiendo crear sociogramas.

- **Unir proyectos:** Esta función permite unir dos proyectos en uno al cual se agregan todos los elementos de ambos.

#### Restricciones

Una característica poco deseable Nvivo es que las versiones varían según el sistema operativo. Esto es un problema para los trabajos colaborativos, ya que si bien

| | Atlas.ti | MaxQDA | NVivo | QDAMiner |
|---|---|---|---|---|
| **Tipo de software** | Propietario | Propietario | Propietario | Propietario |
| **Precio** | 1,840$ | Standard $1140<br>Plus $1380<br>Analytics Pro $ 1800 | Pro $1399<br>Plus $ 1599 | $ 2195 Version con pocas funcionalidades gratis |
| **Prueba gratis** | funcionalidades limitadas | 14 días | 14 días | 30 días |
| **Licencia estudiantes** | $99 – 2años/$51 – 1 semestre | Analytics Pro $ 109 - 1año<br>Standard $ 99 - 2años<br>$ 49 - 1 semestre | 2 años - Pro $99<br>Plus $114 | Def - $ 595<br>1 año $ 238 |
| **Sist.operativo** | Windows, Mac | Windows, Mac | Windows, Mac(limitado) | Windows |
| **Ttipos de archivos** | RTF/D, TXT, DOC/X, PDF | RTF/D, DOC/X, ODT, PDF, TXT, HTML | RTF/D, TXT, DOC/X, PDF | RTF/D, DOC/X, PDF, TXT, HTML |
| **Trabajo colaborativo** | Permite realizar merge de proyectos. Varios usuarios pueden compartir un proyecto, pero no permite el acceso a más de un usuario en simultánea | Permite hacer merge de proyectos, pero no compartir un mismo proyectos netre varios usuarios | Se puede hacer merge de proyectos. Se puede usar un mismo proyecto por varios usuarios en tiempo real sólo mediante la compra deun paquete extra llamado Nvivo Server. | Permite hace rmerge de partes de un proyecto realizadas por disitntos usuarios. Provee una funcionalidad de calificación cruzada entre usuarios del trabajo de los demás para evaluar la confiabilidad del análisis. No permite trabajo conjunto en tiempo real |
| **Restricciones importantes** | - No es posible relacionar los códigos de forma jerárquica.<br>- Un proyecto no puede ser accedido por varios usuarios al mismo tiempo. | Un proyecto no puede ser accedido por varios usuarios al mismo tiempo. | Las funcionalidades incluídas en las versiones varían según el sistema operativo, la versión para MAC no permite funcionalidades importante como por ejemplo el trabajo colaborativo en tiempo real. | En sistemas operativos Linux o MacOS es necesario instalar una máquina virtual para ejecutar el programa |
| **Mejores caracteristicas** | Constructor de modelos y mapas de redes | Análisis con métodos mixtos | Trabajo simultaneo con NVivo Server. | Permite realizar puntuaciones sobre el trabajo realizado por otros usuarios. |

FIGURA 2.1: Tabla comparativa de herramientas.

existe un paquete que permite el trabajo colaborativo en tiempo real, éste está disponible solo para la versión Windows. Además la versión para MAC solamente permite unir proyectos creados con versiones para el mismo sistema operativo.

### 2.1.5. Comparación de herramientas

Se observa que en muchos aspectos estas herramientas son similares, presentando funcionalidades semejantes. En la tabla 2.1 se realiza una comparación entre las características que las diferencian y que son de mayor interés para los objetivos planteados para el desarrollo de Qode. Esta comparación se basa en lo visto en los sitios de las herramientas observadas, los estudios comparativos realizados por las universidades mencionadas al comienzo del capítulo y algunas discusiones en la *web* respecto a este tema[19].

Viendo el anterior análisis realizado de las principales herramientas del mercado, es difícil elegir una como la mejor o líder en el ámbito, cada una se puede adaptar mejor o peor a las necesidades del usuario en función de la metodología de trabajo que este aplique. Algo que sí tienen en común todas ellas es que todas son pagas, con licencias muy altas, lo cual a pesar de haber descuentos para estudiantes genera que sean de difícil acceso a nivel académico. A su vez un usuario que pretende adquirir una de ellas tiene que tomar una decisión muy complicada solo mirando versiones de prueba con pocas de las funcionalidades o con un período muy acotado como para poder evaluar a fondo las distintas opciones. Además de ser pagas, la mayoría de ellas son aplicaciones de escritorio y con *software* privado, lo cual implica que para acceder a las versiones más nuevas se sigue dependiendo de comprar nuevas licencias, siendo así el usuario un rehén de la situación.

[19] predictiveanalyticstoday.com/compare/qda-miner-lite-vs-nvivo-vs-atlas-ti-vs-maxqda/
blogs.warwick.ac.uk/cesphd/entry/atlasti_vs_nvivo

De todas formas existen algunas herramientas gratuitas, una de ellas es LibreQDA, surgido en la Universidad de la República con la motivación de crear una herramienta gratuita y de *software* libre que pueda servir con fines académicos.

## 2.2. LibreQDA

LibreQDA[20] surgió en el año 2012, se analizó qué requerimientos debía satisfacer el programa resultante y se planificó una forma de trabajo en etapas para ir agregando funcionalidades progresivamente. La iniciativa del proyecto surgió de un grupo de investigadores y docentes universitarios de la Universidad de la República (Uruguay) y de la Universitat Autónoma de Barcelona (España). Estos fueron los ideólogos de la herramienta, que mediante la obtención de fondos para invertir en ella contrataron a una empresa privada dedicada al desarrollo de *software* para llevar a cabo la implementación de la misma. Sin embargo, el proceso de desarrollo se estancó a partir del año 2013 y no tiene expectativas de ser retomado.

De todas formas vale la pena observar la herramienta realizada y las decisiones tomadas para aprender del análisis hecho por los especialistas que la idearon, y tal vez aprovechando las características de *software* libre y de código abierto tomar este proyecto como punto de partida. Para esto se contactó a Luis Pablo Alonzo[21] quien nos contó del proceso de LibreQDA, la relación con la empresa de desarrollo y el especial interés que tiene una herramienta libre para el análisis cualitativo, además de expresar su apoyo en caso de que se decidiera continuar con el desarrollo de dicha herramienta.

LibreQDA es una aplicación *web*, es decir, una herramienta para usar en línea. Esto es bueno para fomentar el trabajo colaborativo en tiempo real, y para poder brindar la herramienta sin que el usuario tenga que realizar la instalación (sólo una primera vez en el servidor donde esta se ejecute). La primer impresión que tuvo el equipo con LibreQDA es que su instalación es compleja y poco intuitiva para un usuario normal. Se deben seguir una lista de pasos muy larga sobre todo instalando paquetes o librerías, se tienen los comandos necesarios pero algunos están desactualizados y hay que investigar para resolverlos. Además la instalación es específica para el sistema operativo Ubuntu[22] y las versiones de lenguajes y librerías utilizadas han quedado algo obsoletas.

Analizando el planteo del desarrollo, la etapa 1 consistía en la sistematización de documentos con asignación de metadatos correspondientes, segmentación y codificación de documentos, asociación de anotaciones(memos) a las distintas entidades, recuperación de dichos segmentos con vistas a continuar el análisis con otras herramientas o a la elaboración de informes. Sin embargo, el desarrollo no alcanzó a completar un versión robusta de las funcionalidades definidas en dicha etapa.

El motivo del bloqueo en el proceso de desarrollo fue el descontento del grupo académico, considerando que el resultado obtenido a esa altura del proyecto era muy pobre en comparación con el precio pagado a la empresa encargada de la tarea.

Finalmente luego de evaluar la herramienta existente se optó por comenzar un nuevo proyecto de cero, pero tener contactos con los ideólogos del proyecto LibreQDA para que ellos transmitan su experiencia, y dar aportes de como se usa el *software* en el proceso del análisis cualitativo, temática en la que teníamos nula experiencia.

[20] www.libreqda.edu.uy/

[21] Sociólogo. Defensor del *software* libre. Participante de proyecto LibreQDA

[22] ubuntu.com

Los motivos para desechar la idea de continuar el desarrollo de LibreQDA fueron varios. Teniendo como uno de los objetivos principales dejar una base sólida para que la comunidad abocada al desarrollo de *software* libre, y más específicamente al *software* libre de análisis cualitativo, la obsolescencia de las versiones de librerías y framework era una gran limitante. Esto habría implicado tener que comenzar por actualizar las mismas con versiones actualizadas, labor que insume mucho esfuerzo para un proyecto acotado en tiempo como este.

Otro motivo fue que se consideró que la arquitectura de la aplicación no es la mejor para el funcionamiento de la herramienta y siguiendo la idea propuesta por los impulsores de LibreQDA, de que la interfaz tiene que tener buena respuesta y facilidad de uso para hacer el análisis lo más interactivo posible. Se cree que lo más acertado es agregarle lógica al cliente y que este haga las llamadas al servidor de forma asincrónica, aspecto que se trata más detalladamente en el Capítulo 4.

# Capítulo 3

# Análisis Funcional

En este capítulo se presentan las funcionalidades básicas que una herramienta de análisis cualitativo de datos asistido por computadora debe tener, luego también como extra se añaden funcionalidades deseables pero no básicas (basado en Gibbs, Taylor y Lewins, 2005 y entrevista personal con Alén Pérez, 6 de Junio, 2017). Finalmente se presenta el desarrollo propuesto por el grupo donde se detalla cuales de estas funcionalidades se planea incluir en Qode.

## 3.1. Funcionalidades de una herramienta CAQDAS

### 3.1.1. Funcionalidades básicas

#### Codificación

Codificar implica identificar palabras, frases o incluso párrafos (o fragmentos en caso de imágenes, audio o video) de un documento, que representan cierto concepto. A dichos conceptos se los identifica con un nombre, al que se le llama código o categoría. Lo que se hace al codificar es dejar el fragmento identificado asociado al código pertinente. A medida que se van analizando los documentos, los códigos pueden contener muchos fragmentos distintos pertenecientes a distintos documentos.

Es deseable que la organización de los códigos se pueda crear con forma de árbol, logrando generar especificaciones entre estos.

Herramientas para codificar:

- forma ágil de marcar fragmentos y asociarlos a un código.
- fácil creación de códigos(estructura arbórea para códigos).
- buena visualización del documento que se está codificando, es necesario que la lectura sea cómoda para el usuario.
- visualización intuitiva en el documento de los fragmentos ya asociados a un código.

#### Variables/Atributos

Las variables o atributos son características asociadas al documento. Estos atributos representan información asociada al documento pero que no necesariamente se encuentra contenida en él, como podría ser el nombre del autor, el lugar donde se realizó, el año de realización, etc. Por lo general cada atributo puede tener más de un valor, y cada documento toma un valor para cada uno.

Herramientas para el manejo de atributos:

- administración de atributos: poder crearlos y decidir el tipo de datos aceptan, además de poder modificarlos o borrarlos.
- forma fácil de visualizar y modificar los valores de los atributos para un documento.

**Memos/Anotaciones**

Las anotaciones o *memos* se usan para poder escribir pequeños textos que sirvan como ayuda memoria o documentación del proceso de codificación. Los *memos* pueden ser asociados a un código o a un fragmento de documento, sirven generalmente para que quien está realizando el análisis pueda marcar el motivo por el cual se crea un código específico, o el porqué se asoció determinado fragmento a un código, etc.

Herramientas para manejo de anotaciones:

- fácil visualización de *memos*,pero esta visualización debe ser siempre opcional, ya que los *memos* sirven como documentación pero no son lo esencial en la tarea de análisis cualitativo.
- fácil creación,modificación y borrado de *memos*.

**Búsqueda de texto**

La búsqueda de texto por palabra o frases es una funcionalidad muy útil para cualquier herramienta que se use para estudiar un texto. En un herramienta de análisis cualitativo, donde no sólo se va a estudiar un texto, sino que se va a a hacer un análisis sobre los significados de texto y se va a categorizar fragmentos del mismo, tener esta funcionalidad es deseable.

Herramientas para búsqueda de texto:

- se requiere poder buscar la ocurrencia de palabras, o frases (también pueden ser patrones) en todos los documentos de forma rápida.

**Búsqueda de segmentos codificados**

Las funcionalidades que se provean en esta categoría son muy importantes para el encargado del análisis. Estas funcionalidades son muy útiles para ver los resultados de la codificación realizada. La idea de este tipo de búsquedas es poder ver en qué documentos aparece cada códigos y la relación de aparición entre códigos distintos.

Herramientas para búsqueda de ocurrencias de códigos:

- búsqueda de ocurrencias individuales.
- búsqueda de combinaciones booleanas de códigos(AND : aparecen ambas códigos, OR: aparece uno u otro, etc)
- búsquedas por proximidad: se buscan los lugares donde aparecen más de un código y las búsquedas pueden ser por cercanía en el documento, por estar una ocurrencia seguida de otra, por estar una ocurrencia contenida en otra,etc.

**Modelos gráficos**

Los modelos gráficos sirven para visualizar relaciones conceptuales entre los códigos y/o documentos. Son diagramas que representan relaciones entre elementos claves en el campo de estudio.

Herramientas de modelos gráficos:

- disponibilidad de algunos modelos básicos para visualizar los resultados de la codificación.
- posibilidad de crear nuevos modelos eligiendo las categorías, documento, variables y relaciones entre sí deseadas para la visualización.

**Análisis dimensional**

La idea del análisis dimensional es poder mezclar el análisis cualitativo realizado con la codificación con un análisis cuantitativo en función de los valores de los documentos para cada atributo.

Herramientas para el análisis funcional:

- agrupaciones: poder agrupar resultados según los valores de los documentos para determinados atributos.
- filtrado: poder filtrar resultados según los valores que toman los documentos en ciertos atributos.

**Hipertexto**

Tener la posibilidad de insertar vínculos de hipertexto, permite generar una asociación entre documentos que se referencian entre sí.

Herramienta de hipertexto:

- inserción de hipervínculos de un documento a otro para hacer referencia a una cita.

**Reportes**

Creación de reportes que puedan ser editables por el usuario, pudiendo seleccionar los datos que se quieren mostrar. De gran utilidad para mostrar los datos de mayor interés al realizar análisis dimensional.

Herramientas para reportes:

- tener algunos reportes básicos estándar.
- posibilidad de crear reportes donde el usuario elige que se muestra.

### 3.1.2. Funcionalidades avanzadas

Las siguientes funcionalidades que se describen se llaman avanzadas porque no son parte de lo esencial del análisis cualitativo. Estas funcionalidades se pueden ver como un valor agregado para una herramienta de este tipo.

**Sugerencias de codificación**

Funcionalidad de sugerencia de codificación de fragmentos por relación semántica con fragmentos ya codificados por el usuario. Se aplica en base a un conjunto de reglas de relación semántica entre palabras como podría ser una ontología o un tesauro.

**Todas las búsquedas básicas pero usando relaciones semánticas**

Uso de relaciones semánticas para las búsquedas, tanto para las búsquedas por palabras o patrones, como para las búsquedas por código.

**Auto codificación**

Capacidad de la herramienta de codificar automáticamente. Esto se puede realizar mediante relación semántica de términos y/o aprendizaje automático de la codificación del usuario.

**Conteo de frecuencias de palabras**

Funcionalidad que permite saber con la frecuencia y la cantidad de ocurrencias de palabras en determinado conjunto de documentos. Esto permite hacer análisis de relación entre códigos y palabras, como qué términos aparecen más veces en los segmentos codificados para cada código por ejemplo.

**Trabajo colaborativo**

Capacidad de trabajo en paralelo de más de un usuario sobre un mismo documento o proyecto. En este aspecto hay dos tipos de trabajo colaborativo, concurrente y no concurrente. El caso no concurrente es sencillo de implementar y prácticamente podría ser considerado entre las funcionalidades básicas.

Para permitir concurrencia en la codificación de un documento ya es un trabajo mucho más complicado para el programador, es conveniente realizar un análisis de qué tan productiva es dicha funcionalidad para los analistas cualitativos.

**Atajos de teclado en codificación**

Esta funcionalidad consiste en darle la posibilidad al usuario de acceder rápidamente a la creación de códigos, codificación simple o con códigos activados y la posibilidad de configurar atajos de teclado personalizados para los códigos que crea. De esta forma se le da una comodidad agregada para los usuarios que les gusta poder codificar usando solamente el teclado.

## 3.2. Desarrollo Propuesto

El contexto de este proyecto implica al cliente, el cual realiza distintos análisis cualitativos de datos como ya se presentó en secciones anteriores. Como el cliente no tiene experiencia con herramientas de este tipo se decidió basarse en lo visto al analizar las herramientas líderes del mercado y en las recomendaciones de Alén Pérez (entrevista personal, 6 de Junio, 2017), especialista en este tipo de herramientas y uno de los impulsores del proyecto libreQDA, quien sugirió que lo más importante que debe tener una herramienta para realizar QDA es, primeramente, una interfaz

cómoda para la lectura y codificación de los documentos. Por esta razón se decidió darle prioridad en una primer etapa del desarrollo a generar un ambiente de trabajo cómodo y de fácil interacción para el usuario a la hora de la codificación, para luego en las posteriores etapas ir agregando progresivamente funcionalidades más complejas de búsquedas y visualización de resultados.

El único requerimiento planteado por el cliente es brindar un mecanismo de comunicación para que una herramienta a desarrollar por otro grupo de estudiantes de Proyecto de grado pueda cargar proyectos y documentos de forma directa. Dicho proyecto consiste en un sistema que le permita a la cliente crear las encuestas para sus talleres y le brinde una forma sencilla de tomar notas en las reuniones para que estas queden digitalizadas de forma automática.

A continuación se realiza un listado de las funcionalidades potenciales de la herramienta.

**Codificación**

- Creación y edición de códigos.
- Uso de códigos creados.
- Visualización del código.
- Personalización de atajos de teclado para asignar cierto código.
- Poder ordenar los códigos en forma jerárquica.
- Codificación automática(por medio tesauros, ontologías o incluso aprendizaje automático)

**Memos**

- Crear, editar, borrar. (Códigos)
- Crear, editar, borrar. (Documentos)
- Crear, editar, borrar. (Citas)

**Búsquedas**

- Simple por palabra/s (Ctl+F).
- Simple por expresiones regulares.
- Simple por código
- Lógicas (AND,OR,NOT)
- Proximidad
- Superposición
- Semánticas.

**Atributos**

- Crear, editar, borrar.
- Filtrar documentos por atributo.

**Visualización**

- Nubes de palabras
- Posibilidad de editar.
- Reportes de resultados.

**Hipertexto**

- Interno entre documentos.
- Externo a *web*.
- Externo dbpedia.

**Gestión de Proyectos**

- Crear/Eliminar proyectos
- Importar proyecto.
- Agregar/Quitar usuarios
- Asignar roles
- Versionado

**Gestión de Usuarios**

- Registro
- Login

**Importar Documentos**

- Permitir carga de documentos.

**Módulo cliente**

- API para cargar documentos a un proyecto.
- API para crear proyectos.

**Atajos de teclado**

- Atajos de teclado básicos para ciertas acciones.
- Posibilidad de editar/crear atajos para ciertas acciones

**Sugerencias**

- Codificación.
- Creacion de codigos.

**Auto codificación**

- Aprendizaje automático.
- Ontologías.

**Análisis dimensional**

- Agrupar por atributo.
- Agrupar por código.
- Agregación

**Conteo de palabras**

- Conteo de ocurrencias.
- Conteo de frecuencia en conjunto de documentos.

**Trabajo colaborativo**

- Permitir que más de un usuario edite o gestione un proyecto en simultánea.
- Permitir la visualización de lo que otro usuario está codificando en tiempo real.
- Permitir a varios usuarios codificar sobre un mismo documento de forma concurrente.

A partir del listado anterior, se muestra el cuadro 3.1 el resumen de categorías de funcionalidades básicas y en qué etapas del desarrollo serían incluidas. Asimismo, el cuadro 3.2 muestra lo mismo para las categorías de funcionalidades avanzadas y el cuadro 3.3 hace lo propio con el único requerimiento de la cliente, en estos casos especificando si se incluyen o no, ya que algunas funcionalidades avanzadas no se tienen en cuenta para el desarrollo.

| N | Nombre | Incluido | Etapa |
|---|---|---|---|
| 1 | Codificación | SI | 1 |
| 2 | Variables/Atributos | SI | 2 |
| 3 | Memos/Anotaciones | SI | 1 |
| 4 | Búsquedas de texto | SI | 2 |
| 5 | Búsquedas de código | SI | 2 |
| 6 | Modelos gráficos | SI | 2 |
| 7 | Analisis dimensional | SI | 3 |
| 8 | Hipertexto | SI | 3 |
| 9 | Reportes | SI | 3 |

CUADRO 3.1: Funcionalidades básicas.

Aquellas funcionalidades básicas incluidas en el desarrollo en la primer etapa son las que representan las actividades básicas de cualquier análisis cualitativo por lo que deben estar indefectiblemente disponibles. Luego a medida que avanzan las etapas las funcionalidades que aparecen son menos imprescindibles y más complejas de desarrollar.

| N | Nombre | Incluido | Etapa |
|---|---|---|---|
| 1 | Sugerencias | NO | - |
| 2 | Búsquedas semánticas | NO | - |
| 3 | Auto codificación | NO | - |
| 4 | Frecuencia de palabras | SI | 3 |
| 5 | Trabajo colaborativo | Parcial | 1 |
| 6 | Atajos de teclado | SI | 2 |

CUADRO 3.2: Funcionalidades avanzadas.

| N | Nombre | Incluido | Etapa |
|---|---|---|---|
| 1 | Carga directa de datos | SI | 1 |

CUADRO 3.3: Requerimientos cliente.

Igualmente la elección de las funcionalidades avanzadas traen consigo sus argumentos. Los mismos no son tan evidentes como en el caso de las básicas así que serán detallados a continuación:

**Auto codificación** se entiende como asignación automática de códigos basada en codificaciones previas del usuario y en tesauros que marquen la semántica de las palabras que se encuentran en el documento, incluso se manejó la posibilidad de utilizar aprendizaje automático. Este tipo de funcionalidad se dejó de lado por una razón sencilla, el análisis cualitativo varía según cada investigador, el vocabulario y la semántica del dominio varía según el objeto de estudio y la región de origen de la investigación. Todas estas variantes hacen que la codificación mediante tesauros u ontologías sea muy complicada por la gran variedad de significados distintos que puede tener una palabra o un fragmento de texto según el contexto, y para el caso del aprendizaje automático se hace prácticamente imposible, de hecho se han hecho investigaciones para ver si este tipo de tarea es viable, las cuales han tenido resultados negativos

En conclusión la codificación automática, en cualquiera de las formas comentadas podría darse si realizaramos una herramienta para análisis en un determinado campo temático específico de forma de acotar bastante el vocabulario que se maneja en los documentos, y a su vez se precisan documentos estructurados para evitar incongruencias semánticas.Un estudio sobre el tema se puede ver en la tésis de maestría “Using Natural Language Processing to Support Interview Analysis” de Andreas Kaufmann,Friedrich-Alexander University Erlangen-Nürnberg, 2014.

**Sugerencias y búsquedas semánticas** no se incluyen en el desarrollo propuesto al igual que la auto codificación por su complejidad. Si bien estas funcionalidades son un poco más sencillas de llevar a cabo que la auto codificación, no dejan de ser complejas de desarrollar, además de no ser prioritarias para los objetivos planteados.

# Capítulo 4

# Implementación

Este capítulo describe las características y decisiones de diseño e implementación de Qode. La sección 4.1 recorre la arquitectura del sistema, comparando distintos diseños posibles y argumentando la elección tomada. En la sección 4.2 se enumeran y describen en detalle las entidades principales que componen a Qode. A continuación, la sección 4.3 hace una descripción breve pero ilustrativa del flujo de pantallas y funcionalidades relacionadas en cada una. La sección 4.4 se dedica a describir una de las características más relevantes y críticas del desarrollo, siendo esta la estructuración del texto a presentar en pantalla. Luego, en base a lo explicado en la sección anterior y con un análisis más profundo, en la sección 4.5 se enumeran ciertas limitaciones del sistema. La sección 4.6 analiza todo los aspectos referidos a la seguridad en Qode, incluyendo los procesos de autenticación, autorización y manejo de roles y permisos. Finalmente la sección 4.7 ilustra los distintos entornos de trabajo y cómo trabajar con ellos.

## 4.1. Arquitectura

### 4.1.1. Escritorio vs *Web*

Las opciones existentes para el diseño e implementación de un CAQDAS son muchas y variadas. En el relevamiento se observó que la mayoría de las herramientas son aplicaciones de escritorio, las cuales por lo general implican un ambiente de trabajo propio de cada investigador. Los documentos a analizar y el propio trabajo sobre los mismos queda guardado en el ordenador del usuario volviendo difícil el trabajo colaborativo; compartir análisis y conclusiones entre investigadores implica un esfuerzo agregado y poco eficiente. También existe complejidad en relación al funcionamiento de estos programas en base a las especificaciones de *hardware* de cada computadora. Es común en análisis cualitativo tratar con grandes volúmenes de texto los cuales, deberán ser procesados en memoria principal y también guardados en memoria secundaria. Sin embargo, los CAQDAS de este tipo tienen algunas ventajas, como ser siempre estar disponibles para utilizar y que la confidencialidad de los datos queda bajo la responsabilidad única del investigador.

Ventajas de opción **Escritorio**

- Contínua disponibilidad.
- Mayor asegurabilidad de los datos confidenciales.

Desventajas de opción **Escritorio**

- Complejidad para trabajar en forma colaborativa.

- Limitaciones por características de *hardware*.
- Los cambios de versión son difíciles de realizar.

En contraste con las aplicaciones antes mencionadas, otra opción es la de desarrollar el sistema como una aplicación *web*. Las mismas existen en diversos tipos, lenguajes y variantes arquitectónicas, pero en rasgos generales son aplicaciones accedidas a través de un navegador *web* que siguen la arquitectura de cliente-servidor. Las propias variantes de esta arquitectura definen distintas ventajas y desventajas para este tipo de sistemas. Las ventajas de una aplicación de cliente *web* fino o ligero[1] no serán las mismas a unas de cliente grueso o pesado[2]. Sin embargo en contrapartida con las aplicaciones de escritorio y en relación a los CAQDAS se puede mencionar algunas características generales que se comparten.

En primer lugar el acceso vía *web* admite que varios investigadores trabajan sobre los mismos documentos, pudiendolo hacer simultáneamente o no. Esto favorece la estandarización de nomenclaturas y la eficiencia del trabajo. A su vez los cambios de versión serán transparentes para los usuarios y todos se beneficiaran de los mismos inmediatamente, ya que no habrá esperas para bajar parches o problemas de instalación de los mismos. El volumen de documentos será responsabilidad del servidor donde sean guardados por lo que la capacidad de disco de cada computadora tampoco será una limitante (la capacidad de memoria rápida será relevante según el tipo de arquitectura *web*).

Por otro lado los datos confidenciales quedarán en servidores ajenos al investigador por lo que la responsabilidad de estos datos recaerá en quien provea el servicio. También alguna falla de red implica que la aplicación quede inutilizable durante ciertos períodos.

Ventajas de opción **Web**

- Trabajo colaborativo.
- Trabajo simultaneo.
- Facilita la estandarización de nomenclaturas.
- Fácil transición entre versiones.
- Parcialmente independiente del *hardware* utilizado por el usuario.

Desventajas de opción **Web**

- Datos confidenciales en el servidor.
- Disponibilidad dependiente de la red.

No es menor aclarar que cada gran paradigma arquitectónico trae incorporado una enorme cantidad de variantes que también han de ser comparadas para el diseño final. De cualquier manera la decisión de utilizar una aplicación en el formato de escritorio o *web* es una de las decisiones principales a tomar antes de comenzar el desarrollo.

[1] El procesamiento de la información y la gestión de los datos ocurre en el servidor. El cliente sólo es responsable de ejecutar el *software* de presentación.(Sommerville, 2005)

[2] El servidor es responsable sólo de la gestión de los datos. El cliente ejecuta el procesamiento de la aplicación (la lógica de la aplicación) y la presentación.(Sommerville, 2005)

Teniendo en cuenta los aspectos antes mencionados y persiguiendo fielmente el objetivo de desarrollar una aplicación libre, escalable y práctica se optó por perseguir el camino *web*. Se puede interpretar que las ventajas de estas, superan a las de las versiones de escritorio. Además las desventajas mencionadas pueden ser mitigadas con buenas medidas de seguridad y robustez en la red de trabajo.

### 4.1.2. ¿Cual arquitectura *Web*?

El siguiente paso hacia determinar la arquitectura que finalmente se implementará es elegir que tipo de aplicación de *web* es la mejor opción. Esto refiere a las variantes hoy existentes en el mercado desde el punto de vista arquitectónico de estos sistemas. Inevitablemente estos aspectos van acompañados de tecnologías particulares las cuales ineludiblemente se debieron investigar y contrastar para tomar la decisión final.

Actualmente la mayoría de las aplicaciones *web* implementan el modelo de arquitectura en tres capas (Sommerville, 2005). La arquitectura clásica de este tipo consta del navegador como interfaz del usuario y presentación de datos, la aplicación y la base de datos. En un buen diseño se espera que la aplicación y base de datos se encuentren en servidores distintos. En esta arquitectura la lógica de aplicación puede ejecutarse totalmente del lado del servidor donde corre la aplicación, teniendo así lo que se denomina un cliente fino donde el navegador simplemente muestra páginas HTML estáticas y no ejecuta lógica alguna (más allá de lógica simple en botones u otros elementos de la presentación). En el punto opuesto se encuentran los clientes gruesos, los cuales, mediante el uso de JavaScript[3], entre otros lenguajes, traslada la lógica de negocio al navegador y el servidor pasa a ser un simple nexo entre la aplicación y los datos. Existen también puntos intermedios en los cuales la lógica de aplicación se comparte entre navegador y servidor.

En el caso de un CAQDAS hay que primero analizar el funcionamiento de los mismos para luego, en el contexto *web*, optar por cuál diseño es más apropiado. Las actividades fundamentales que se han presentado previamente en las herramientas para análisis cualitativo se pueden resumir en:

- Lectura de documentos
- Codificación
- Búsquedas
- Anotaciones

La codificación se realiza totalmente sobre los objetos en pantalla. En el caso de textos se selecciona porciones del mismo y se debe esperar un comportamiento inmediato para una buena experiencia del usuario. Esto argumenta a favor de tener gran parte de la lógica de aplicación en el cliente. Por otro lado las búsquedas puede ser una actividad pesada si la cantidad de datos a procesar es grande, por lo que podría ser candidato a ser algo a realizar desde el servidor. Sin embargo, los navegadores son hoy en día son cada vez más potentes, por lo tanto no sería, a priori, imposible imaginar que las búsquedas se pudieran realizar desde el propio cliente.

Otro factor a favor de utilizar un cliente grueso es la naturaleza del problema y de los datos. A diferencia de aplicaciones empresariales, por ejemplo, no hay una lógica de negocio compleja o crítica que deba implementarse. En cambio, los CAQDAS

[3] www.javascript.com/

son una herramientas que facilitan el trabajo de extracción y análisis de información, por lo que optar por un servidor que simplemente sirva y guarde información es algo también plausible. A su vez utilizar lenguajes como JavaScript permite realizar llamadas asíncronas al servidor por lo que las porciones de texto seleccionadas para codificar, por ejemplo, se guardarán en la base datos sin que el usuario perciba demora alguna. Al estar gran parte de la lógica en el cliente se da que la velocidad de respuesta es muy buena lo que mejora la experiencia del usuario.

El análisis anterior deja, en gran medida, la certeza que utilizar un cliente grueso es la más adecuada para un CAQDAS en un entorno *web*. De cualquier manera no se deben dejar de lado algunas cuestiones que pueden ser claves para el éxito de este diseño. Existen un aspecto que a priori no se puede determinar con exactitud el efecto que puede tener en un sistema desarrollado bajo este modelo, pero que puede afectar el buen funcionamiento de la aplicación en producción. Es posible que algún investigador pretenda usar textos realmente largos, por ende archivos de gran tamaño, los cuales deberán ser procesados al momento de la carga y luego mantenidos en memoria mientras se interactúa con estos. Además, teniendo en cuenta que la memoria disponible no es toda la existente en una computadora, sino que aquella asignada al proceso del navegador, es de esperar que este factor pueda ser un cuello de botella, ya que un documento voluminoso puede interferir con el la performance del sistema. Se deben tomar medidas en la implementación para prevenir estas cuestiones, opciones como limitar el tamaño de los archivos y la cantidad de archivos abiertos simultáneamente pueden ser solución a la problemática planteada. Además este tipo de parámetros puede utilizarse para testear los tiempos en las búsquedas y ajustarlos de tal forma que la experiencia del usuario sea lo mejor posible. En la sección 4.5 se trata este tema en profundidad.

### 4.1.3. Base de Datos

Donde guardar la información es otra decisión de diseño que se debe tomar. Las bases relacionales son siempre el primer candidato a tener cuenta. Sin embargo existe una cuestión en la propia naturaleza de los datos a utilizar, que podrían volver a las bases no relacionales más eficientes, en particular se hará enfoque en las documentales. Para determinar cuál utilizar se debe tener en cuenta que modelo se adapta mejor a los requerimientos de la aplicación a desarrollar. Se analizan a continuación sus características principales.

Las bases de datos NoSQL[4] no utilizan el modelo relacional, los datos almacenados no requieren estructuras fijas como tablas. Una base de datos de documentos es una base no relacional que almacena datos como documentos estructurados, típicamente XML y JSON (*¿Qué son las bases de datos documentales?* 2017, Kyocera). La estructura de un documento es simple y compuesta por pares clave/valor, de forma similar a los objetos utilizados en los lenguajes de programación, lo cual permite un mapeo prácticamente directo entre ellos (*¿Qué es una base de datos documental?*, Amazon). En un documento, se pueden agregar, eliminar, modificar o renombrar nuevos campos en cualquier momento lo cual proporciona escalabilidad y garantiza un alto nivel de flexibilidad. Por otra parte, en un modelo relacional para realizar un cambio de modelo se debe alterar el esquema de las tablas.

Las bases de datos relacionales tienden a funcionar más lentamente cuando en una tabla se encuentran cantidades muy significativas de registros lo que obligaría a tomar medidas como ser la división de las tablas (Baquero, 2016). Los archivos

[4]Castro Romero, González Sanabria y Callejas Cuervo, 2012

subidos al sistema (docx, pdf, txt, etc) no serán persistidos en la aplicación, sino que se mantendrá una copia del contenido de los mismos, por lo cual el tamaño de cada documento (base) no será a priori significativo para que se observe una reducción de performance por esta causa. Además, la mayoría de las operaciones que se realizan son de tamaños considerablemente pequeños, a excepción de la inserción u obtención de los documentos (informes, entrevistas, reuniones ). Sin embargo, un modelo relacional podría ser mejor para el análisis posterior donde se estudian las relaciones entre los datos. En la sección 4.5 se hace un análisis más minucioso de los problemas enfrentados para la mejora de la performance.

En el modelo relacional se elimina la redundancia en los datos mediante la normalización, asegurando que se cumplan las dependencias y restricciones entre los datos lo cual, en comparación, las hace más fiables. A pesar de ser estas grandes ventajas, las características de la aplicación hacen que no sea necesario la verificación de la consistencia de los datos por parte de la base de datos, quedando esto bajo responsabilidad de la implementación realizada.

Se concluye, por todo lo anterior que una base de datos documental es la más idónea, dado que los datos que se utilizan son poco estructurados y se actualizan en tiempo real, realizando muchas operaciones pequeñas en un corto tiempo. Con esta implementación se tiene la posibilidad futura de escalar horizontalmente, modificar la estructura de los modelos creados o crear nuevos sin necesidad de una gran reestructura.

Se decide utilizar MongoDB[5], una base de datos orientada a documentos de elevado rendimiento y código abierto. Esta base de datos se utiliza mucho en la industria, contando con implantaciones en empresas como Foursquare [6] y LinkedIn [7]. Plataformas de formación como Codecademy [8]. Otras son eBay [9] o IBM [10], entre otras.

### 4.1.4. Lenguajes y *Frameworks*

Inmediatamente de elegir la arquitectura a utilizar, sigue la determinación de qué tecnologías serán las cuales permitirán la forma más óptima realizar el desarrollo esperado. La decisión de basarse en una aplicación *web* con fuerte procesamiento del cliente lleva, naturalmente, a volcarse al uso de algún framework y/o lenguaje que facilite este trabajo. En la actualidad las denominadas aplicaciones single page son sistemas desarrollados sobre el concepto de cliente grueso, las cuales se basan en algún lenguaje que puede ser interpretado por los navegadores más comunes, generalmente Javascript. Existen en estos días dos tecnologías que lideran este tipo de desarrollos, Angular [11] y ReactJs [12]. El primero creado por Google [13] y el segundo por Facebook [14], y ambos utilizados en desarrollos propios de ambas empresas.

Los debates (*5 Best Javascript Frameworks In 2017* 2017)(*Javascript Frameworks you should know* 2017)(Neuhaus, 2017) sobre cuál framework es mejor, es extenso en parte, por ejemplo, porque React no se considera como framework (*Is React library or*

[5] www.mongodb.com
[6] es.foursquare.com
[7] linkedin.com
[8] www.codecademy.com
[9] www.ebay.es
[10] www.ibm.com
[11] angular.io
[12] reactjs.org
[13] google.com
[14] facebook.com

*a framework?* 2016)(*Is React a Framework?* 2015), entre otros aspectos. De cualquier manera para la elección de que utilizar para el desarrollo del CAQDAS se opta por utilizar Angular en su versión 4. Las razones son varias, una de ellas es que Angular tiene *"Dependency injection"*[15] que básicamente implica que se puede tener objetos provistos como servicios para otros facilitando la aplicación del patrón de diseño MVC, ampliamente utilizado en el contexto de aplicaciones *web*,y que aplica para esta herramienta en la cual el modelo es muy importante. Usar este patrón de diseño le da más claridad al código, un aspecto muy deseable para una herramienta de código abierto y escalable. Otro motivo para elegir Angular es que es un *framework* completo, mientras que React, como se mencionó anteriormente, es una librería. Esto quiere decir que el primero es un paquete con todas las necesidades, lo cual si bien da menos libertades porque es más rígido en cuanto a que clases o paquetes se debe usar, también facilita la toma de decisiones ya que hay que no hay que preocuparse sobre problemas de compatibilidad. Otra facilidad que vuelca la balanza a favor de Angular para el caso de nuestro proyecto es que este framework ofrece *"Two Way binding"* que básicamente implica que los componentes de la UI están directamente relacionados con propiedades del modelo, lo cual implica poder diseñar más rápido la interfaz al tener que realizar menos configuraciones para los eventos de generados por el usuario, y además supone mayor velocidad de respuesta para el usuario, esto se adapta a las características interactivas de las tareas a realizar en nuestro programa.

Al otro extremo de la aplicación se encuentra la fuente de datos. En su vasta mayoría las aplicaciones *single page* se comunican con alguna API, la cual les permite simular las operaciones CRUD [16] de inserción, modificación, borrado y consulta de datos. Estas APIs pueden estar desarrolladas en lenguajes totalmente diferentes, sobre infraestructuras disímiles y ser propiedad de distintas entidades. Lo imprescindible es que basen la comunicación de información en formatos como XML o JSON, de esta forma independientemente del lenguaje del sistema que realiza la solicitud la API lo puede interpretar y viceversa.

Es entonces que es necesario también elegir qué tecnología se utiliza del lado de la API para Qode. Para dicha elección se tiene en cuenta la base de datos ya que es con esta con la que la API deberá comunicarse constantemente para manipular los datos. Nuevamente la variedad de abundan las tecnologías que se podrían elegir, pero existe un factor más que incide en esta decisión; a pesar que la carga de procesamiento se piensa, a priori, realizar totalmente en el cliente hay una posibilidad de que sea necesario delegar cierto trabajo al servidor. Es una realidad que en la definición purista de una API, esta no debería de realizar más trabajo que cumplir con las operaciones CRUD, pero la naturaleza de los datos puede incidir en optar por romper con dicha definición (Mikowski, 2015). Concretamente, es de suponer que en versiones avanzadas del sistema se utilice gran variedad de procesamiento de lenguaje natural. Aunque gran parte del mismo es posible implementarlo en el cliente utilizando Javascript es una realidad que funcionalidades basadas en aprendizaje automático (ML) u otros algoritmos complejos de NLP sería complejo realizarlo en el cliente por los recursos necesarios.

Aunque no es algo central de un CAQDAS y pese a que en el alcance de este proyecto no se incluyen funcionalidades de este tipo, con el objetivo de que este sistema puede continuar siendo desarrollado hay que prever desde el diseño inicial como facilitar el escalamiento. En la actualidad, todo lo relacionado al procesamiento

[15] angular.io/guide/dependency-injection
[16] restapitutorial.com/lessons/httpmethods.html

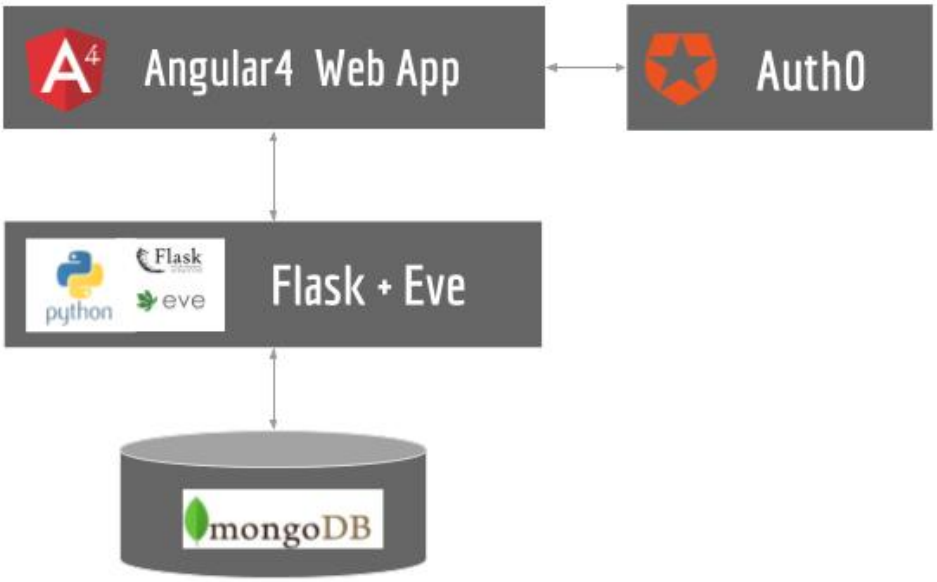



FIGURA 4.1: Diseño arquitectura

de lenguaje natural y aprendizaje automático tiene como líder al lenguaje Python (Voskoglou, 2017)(Puget, 2016). Por lo tanto se opta por buscar algún framework que permita el desarrollo de una API en Python. Resumiendo entonces la elección de qué tecnología utilizar en la API queda restringida a que pueda comunicarse fluidamente con la base MongoDB y que esté desarrollado en Python.

Eve es un *framework* en Python, de código abierto y cuyo uso para el desarrollo de APIs REST va creciendo según el sitio oficial [17]. Eve fue creado para facilitar la creación servicios de tipo REST. Se apoya en dos *microframeworks* Flask [18],el cual sirve como servidor de aplicaciones *web*, y Cerberus[19] ,utilizado para validación de datos. El uso de este *framework* permite implementar la API necesaria que comunique la aplicación Angular con los datos en la base de datos MongoDB. Eve proporciona soporte nativo para base esta base de documentos. El soporte para otras bases de datos está dado por extensiones creadas por la comunidad. Si bien EVE es un *framework* diseñado para crear APIs de tipo REST exclusivamente, al estar basado en otro para aplicaciones *web* normales se heredan todas sus funcionalidades, lo cual garantiza que de ser necesario realizar cierto procesamiento del lado del servidor esto está cubierto por Flask.

### 4.1.5. Diseño Final

Concluyendo esta sección se presenta la arquitectura (Figura 4.1) de Qode. La mayor parte de la lógica de la herramienta se encuentra en el *front-end* que es el único componente que se comunica con el servicio externo de autenticación Auth0[20] (Ver sección 4.6). Para el componente de *back-end* se usa el *framework* Eve, basado en Flask, programado en Python. El mismo se encarga de la interacción con una base

[17] python-eve.org
[18] flask.pocoo.org
[19] docs.python-cerberus.org/en/stable
[20] auth0.com/

de datos MongoDB, donde se almacenan los recursos, y de algunas operaciones que pueden resultar poco performantes en el *front-end*.

## 4.2. Entidades Principales

### 4.2.1. Documentos

Los Documentos son la entidad básica en el sistema, siendo la fuente de información que será codificada y analizada. Cada uno es una representación de un archivo que es importado a la herramienta, la cual recupera el texto asociado simplemente como una cadena de caracteres. El resto de metadatos asociados y la extensión específica del archivo son ignorados, creándose nuevos metadatos asociados a esta nueva entidad. En otras palabras, no importa el tipo de archivo de texto (dentro de aquellos soportados por el sistema para ser importados) la representación final del mismo es equivalente.

Aprovechando las cualidades de MongoDB, los nuevos documentos son guardados no solo con su información propia (texto y metadatos), sino que se embeben otras entidades o referencias a las mismas (*Model One-to-Many Relationships with Embedded Documents*, MongoDB). Las citas serán presentadas a continuación, pero se puede anticipar que tienen su estructura propia y que ocuparan su propio lugar en la base. Dado que estas se componen principalmente por porciones de texto del documento, existe entonces una relación intrínseca entre las citas y los documentos. Este vínculo se optó por modelar en forma referencial (*Model One-to-Many Relationships with Document References*, MongoDB) en la estructura del documento. Se podría haber optado por el camino embebido, pero MongoDB tiene una restricción con el tamaño de los documentos (en referencia a los documentos de Mongo) los cuales no pueden superar los 16 MB (*Documents*, MongoDB). Dado que el propio texto del documento puede llegar a ser voluminoso, incluir dentro del mismo a su vez copias parciales del mismo significa que el tamaño final se vería limitado no solo por la capacidad propia de la base de datos, sino por la cantidad de citas que contenga. La opción referencial es entonces más óptima.

Por otro lado los Memos tienen una estructura y tamaño más acotados. Es así que esta entidad no supondría un conflicto a la hora de embeberla en otras, por lo cual se agregan a la estructura de los documentos.

### 4.2.2. Citas

La codificación implica seleccionar porciones de texto las cuales tienen algún significado particular para el investigador. Esta selección se denominó Cita. Esta es una de las entidades fundamentales ya que vincula la información cruda del documento con la interpretación que el investigador le da. Permite asociar a la misma uno o varios códigos, como también agregar un memo que haga de aclaración o descripción para esa selección del texto.

Con respecto a la persistencia de las citas, como se mencionó en la descripción de los documentos, no es viable agregarla de forma embebida dentro de los mismos. La razón principal es que cada cita mantendrá el texto que la vincula, por lo que el tamaño de las mismas puede no ser despreciable. Si se guardara la cita embebida, el documento de MongoDB que represente a la entidad documento crecería muy rápido y como este tamaño está limitado, la cantidad posible de citas a asociar también quedaría limitada. Por lo tanto, se optó por guardarlas como entidades propias en la base de datos y referenciarlas en cada documento.

Las citas ademas guardan cierta información adicional, como ser la línea en la que inicia y finaliza, como también el caracter en el cual inicia dentro de la primera línea, como en el cual finaliza en la última.

### 4.2.3. Códigos

El fin de la investigación cualitativa es sacar conclusiones a partir del análisis de texto o material audiovisual. Se dispone a relacionar conceptos, para así luego poder formar teorías más abstractas. Los Códigos son entonces el cimiento de estas relaciones.

Los códigos existen transversalmente dentro de un mismo proyecto. Una cantidad **m** de códigos pueden ser asociados a un número **n** de citas dentro de diversos documentos. Estos se identifican con un nombre, pueden tener un color asociado y, al igual que el resto de las entidades antes presentadas, puede tener asociado un memo. Estas características vuelven al código una entidad relativamente independiente, entendiendo esto como que puede existir por sí misma. Cuando una cita es creada es, por lo general, asociada a uno o varios códigos. quedando así formado la relación de estos con con la porción del texto del documento que está siendo analizado.

En MongoDB los códigos se representan como una colección propia, cuyas entidades son referenciadas por las distintas citas. El esquema referencial se elige sobre el embebido por la relación **m**x**n** antes descrita. Aunque igualmente podría utilizarse la segunda opción es una realidad que los códigos son entidades que suelen ser modificadas, por lo que volvería complejo el proceso de actualización de los mismos ya que debería actualizarse cada instancia existente dentro de las distintas citas.

### 4.2.4. Sistema de Códigos

Los códigos presentados en el punto anterior son un componente clave del sistema, el cual vincula las citas con conceptos abstractos determinados por el usuario. Sin embargo, también es posible encontrar una relación propia entre los códigos existentes los cuales permitan al investigador formar conceptos más complejos a partir de ellos. Por lo tanto, el sistema da la opción de organizar los códigos en una estructura jerárquica de forma arborescente. Existe un nodo raíz el cual no tiene ningún código asociado, pero que tiene una lista de nodos hijos los cuales cada uno tiene un código vinculado y su propia lista de hijos.

El sistema de códigos es una representación visual del sistema, es decir en la base de datos los códigos no se guardan de forma jerárquica sino como una simple lista. Esto simplifica la persistencia de los códigos en la base de datos. De cualquier manera de alguna forma hay que guardar la estructura del árbol, la misma se guarda como un un árbol cuyos nodos son referencias a códigos con una lista de hijos que contienen nodos iguales. De esta forma la estructura del árbol se guarda paralelamente a los códigos. Esto permite reducir los tiempos a la hora de crear la representación del árbol en memoria y evita cálculos complejos de posiciones en un árbol a la hora de agregar, modificar, o borrar códigos. Dicho cálculo se realiza únicamente en el árbol mostrado en el navegador y luego se guarda su estructura en la base de datos.

### 4.2.5. Memos

Conceptualmente una de las entidades más simples de la aplicación pero con un rol fundamental tanto en las etapas de codificación como en la de análisis. Los Memos son simplemente cadenas de caracteres asociado tanto a documentos, códigos o citas. Dentro de la base de datos siempre aparecen embebidos dentro de otro objeto.

### 4.2.6. Proyectos

Cada actividad del sistema se realiza dentro del contexto de un Proyecto. Estos permiten diferenciar los trabajos entre diferentes usuarios y a su vez cada uno puede mantener una colección de sus proyectos. Esto implica, a nivel de arquitectura y de base de datos, que los proyectos existan por sí mismos y todos los objetos ya presentados tendrán una referencia al proyecto que pertenecen. Por cada uno se puede tener un número ilimitado de documentos y de códigos. También las citas realizadas sobre los documentos quedarán asociadas al proyecto. A su vez se tendrá un campo que identifique al usuario dueño del mismo. Este vínculo de cada objeto con su proyecto correspondiente facilita la restricción de acceso de distintos objetos entre los usuarios del sistema, este punto se trata en la sección 4.6.

Para modelar que los proyectos sean colaborativos, se debe mantener un conjunto de usuarios "colaboradores". El acceso a estos proyectos es restringido en función del rol que tenga colaborador, como se detalla en la sección 4.6, pero mayoritariamente podrán importar documentos, codificar o realizar búsquedas como en sus propios proyectos. La eliminación de un proyecto implica el borrado de todos los documentos, códigos, citas y por ende, memos asociados al mismo.

### 4.2.7. Usuarios

A diferencia del resto de las entidades descritas los Usuarios no ocuparán un lugar propio en la base de datos. En cambio se optó por utilizar el modelo denominado *Authentication-as-a-Service*. En la sección 4.6 se explica cómo se realiza la autenticación utilizando la infraestructura Auth0, pero cabe aclarar que a nivel de arquitecura un usuario será dueño de una cantidad de proyectos en base a su *e-mail* el cual lo identifica en el sistema. A su vez podrá participar como colaborador en otros proyectos, también siendo identificado en estos por su correo electrónico, siendo dicho permiso otorgado por el creador del proyecto o un colaborador del mismo (con los permisos correspondientes). Para los colaboradores se tendrán dos tipos de acceso: un rol de solo lectura, que restringe cualquier operación sobre el proyecto (creación, modificación ,borrado) y un rol de lectura/escritura que mantendrá permisos totales al igual que el creador con excepción del borrado del proyecto.

La Figura 4.2 resume lo presentado en un modelo de dominio del sistema. Se puede visualizar los distintos atributos de las entidades y las relaciones entre ellas.

## 4.3. Flujo del Sistema

Presentada la estructura de Qode, es importante definir cuál será su comportamiento. En el Capítulo 3 se presentaron los requerimientos funcionales de la herramienta. Es a partir de estos que se define el flujo del sistema. En esta sección se pretende describir las pantallas principales que el usuario se encontrará en el uso de

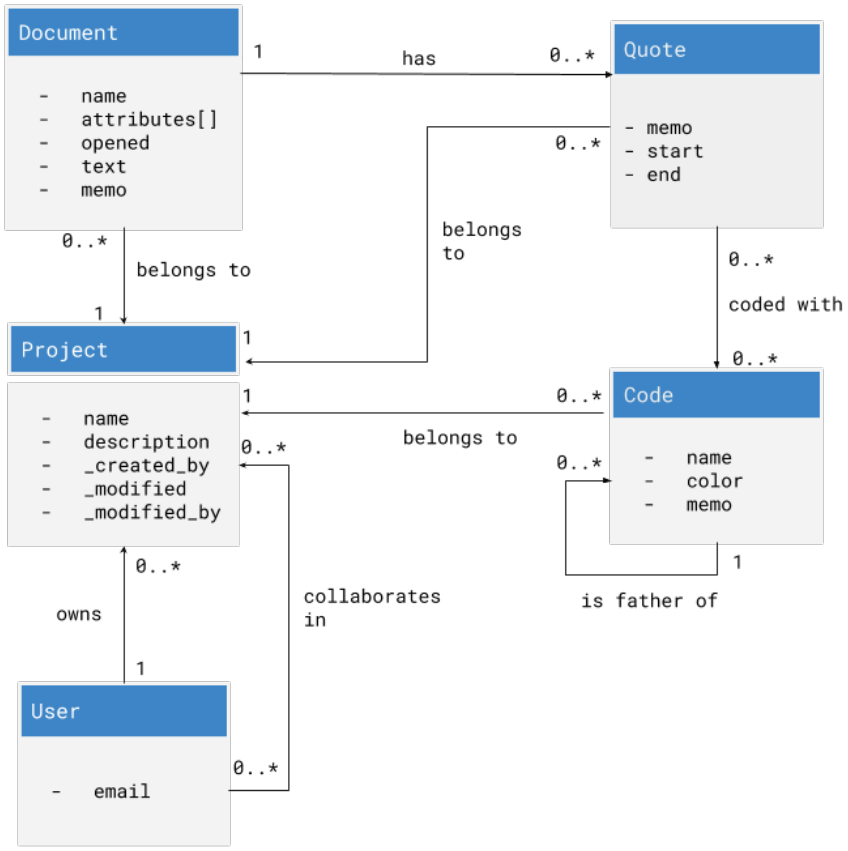



FIGURA 4.2: Modelo de dominio Qode

la aplicación. No es el objetivo describir detalladamente cada una de las funcionalidades a implementar, pero si darle al lector una visión general de la herramienta. En el Apéndice C se puede encontrar el Manual de Usuario, el cual recorre una a una las distintas funcionalidades que la herramienta provee en cada una de las pantallas presentadas a continuación.

Como se explicó anteriormente en este capítulo, se optó por un diseño de cliente fino. El mismo supone que no exista el *refresh* del navegador que ocurre cada vez que el usuario cambia de pantalla o realiza alguna acción que precisa de interacción con el servidor. En cambio, en esta arquitectura la página HTML generada en la primera llamada al servidor es la única existente. Angular provee mecanismos, con el uso de componentes[21] y manejo de rutas [22], que genera código JavaScript que simula el cambio de pantallas sin necesidad de solicitar nada al servidor. Por lo tanto la única interacción con este último ( luego de cargada la página por primera vez) son las llamadas asincrónicas para obtener o guardar datos.

Existen únicamente tres pantallas en el sistema y se acceden en un orden establecido. Las dos primeras implementan las funcionalidades de manejo de usuarios y gestión de proyectos. La última es donde realmente se realiza el trabajo y se encuentran la mayoría de las funcionalidades encontradas en cualquier CAQDAS. A continuación se presenta una descripción breve de cada una y las funcionalidades básicas que proveen.

### 4.3.1. Inicio de sesión o Registro

Al ingresar al sistema se pide al usuario autenticarse (Figura 4.3). Existe la posibilidad de utilizar las autentificación con las credenciales de Google [23] o con el usuario y contraseña clásico. Para utilizar este último método, previamente se debe de registrar el usuario. Tanto el inicio de sesión como el registro de usuarios se encontrarán en esta primera pantalla. Es importante recalcar que no se puede acceder a ninguna otra pantalla o a cualquier recurso del servidor sin antes verificar la identidad del usuario. Esto se realiza mediante el uso de los servicios de Auth0 y JWT *tokens*, los cuales se presentan la sección 4.6.

### 4.3.2. Gestión de Proyectos

Un registro o inicio de sesión válido traslada al usuario a la pantalla de gestión de proyectos (Figura 4.4). En esta se pueden crear nuevos proyectos, ingresando un nombre y una descripción opcional. Se listan también en pantalla los proyectos creados previamente por el usuario autenticado y también todos aquellos en los que colabora. Los permisos en estos últimos serán determinados por aquel que lo invita a colaborar. El manejo de permisos se explica en la sección 4.6.

Al seleccionar un proyecto del listado se puede ver información más detallada, con posibilidad de editar su descripción y compartir el proyecto con otros usuarios. Finalmente, habiendo seleccionado un proyecto se puede acceder al ambiente de trabajo o *workspace* del mismo.

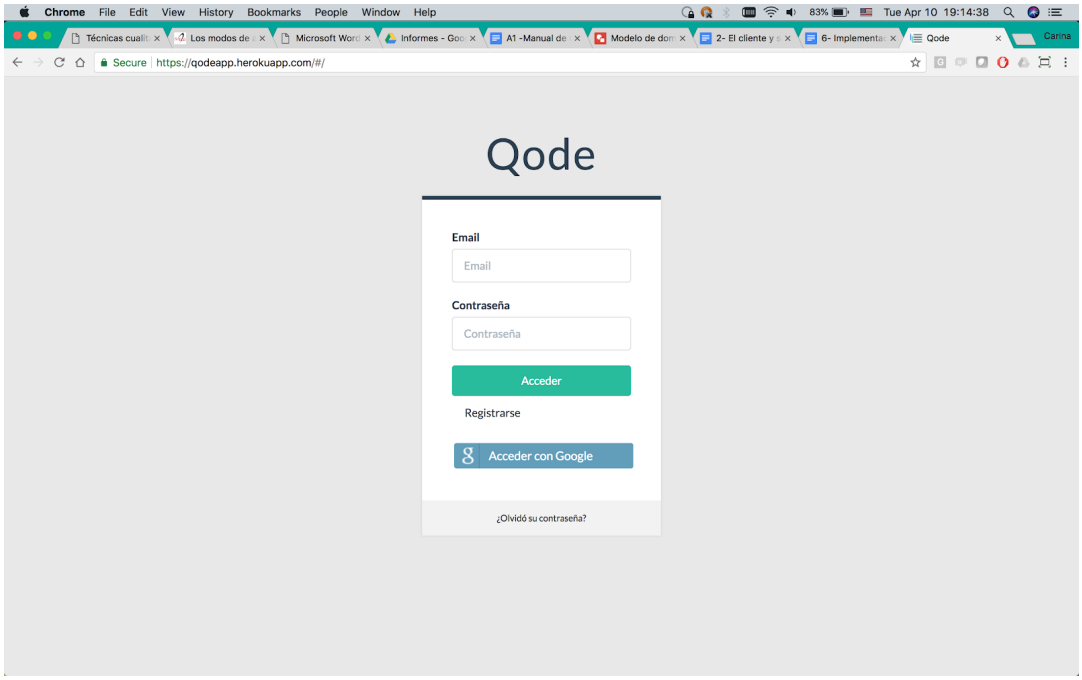



FIGURA 4.3: Página de login

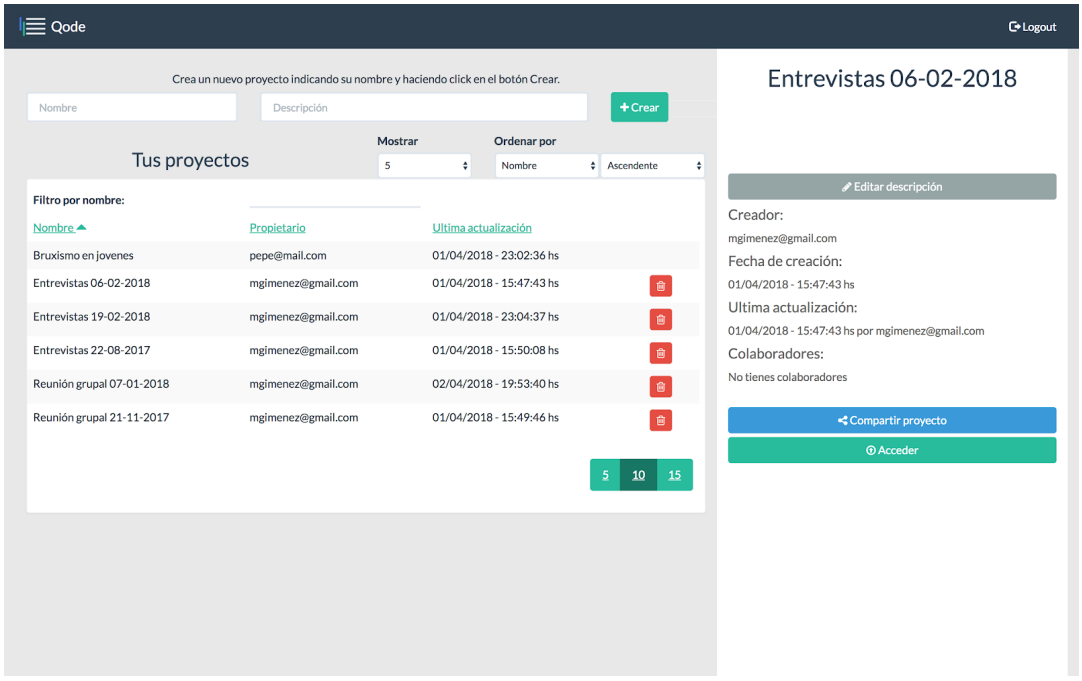



FIGURA 4.4: Gestión de Proyectos

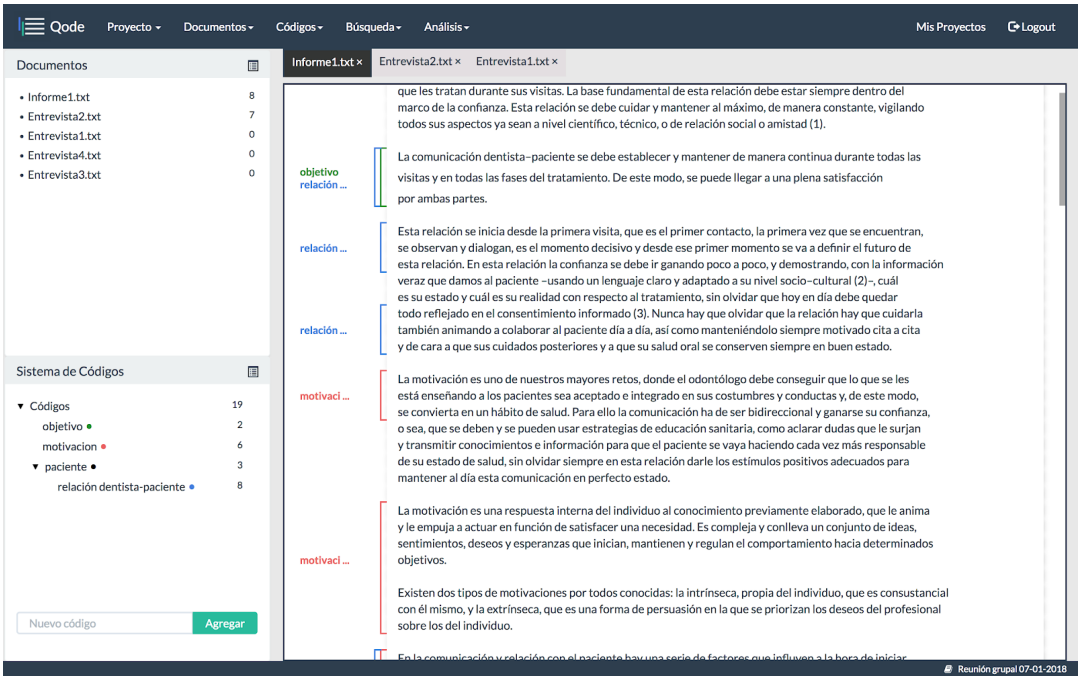



FIGURA 4.5: Workspace

### 4.3.3. Ambiente de Trabajo

Esta pantalla es la más importante de las tres. Es aquí donde el investigador puede realizar el análisis cualitativo. Es una decisión central del proyecto lograr que todo lo relativo al trabajo del investigador pueda ser realizado totalmente en este lugar. El evitar todo tipo de recargas que generen demoras, u otros retrasos debido al manejo de datos, son minimizados utilizando esta única pantalla y las llamadas asíncronas al servidor.

Desde aquí el usuario puede importar documentos de texto (en los formatos permitidos). También tiene la posibilidad de crear o importar códigos y organizarlos dentro del sistema de códigos. Este último está siempre disponible en la barra lateral izquierda, junto al conjunto de documentos importados.

En la parte central de la pantalla, se encuentra el texto de cada documento. Los mismos quedan organizados en pestañas, las cuales el usuario puede abrir, cerrar o navegar. El contenedor del texto abarca la parte más amplia de esta pantalla, permitiéndole al usuario mantener una lectura simple y clara. Además en esta misma sección se puede marcar distintos segmentos de texto, creando así diversas citas asociadas a códigos. Dichas citas se presentan en forma de corchete del lado izquierdo del texto. En la Figura 4.5 se puede observar un ejemplo de un ambiente de trabajo con un texto importado y con algunas citas agregadas.

Este último punto descrito implica un trabajo de diseño considerable y un desafío para el grupo a la hora de implementar la presentación del texto y las citas en pantalla. En la sección 4.4, a continuación, se realiza un análisis en detalle de este punto.

[21] angular.io/guide/architecture-components
[22] angular.io/tutorial/toh-pt5
[23] developers.google.com/api-client-library/javascript/features/authentication

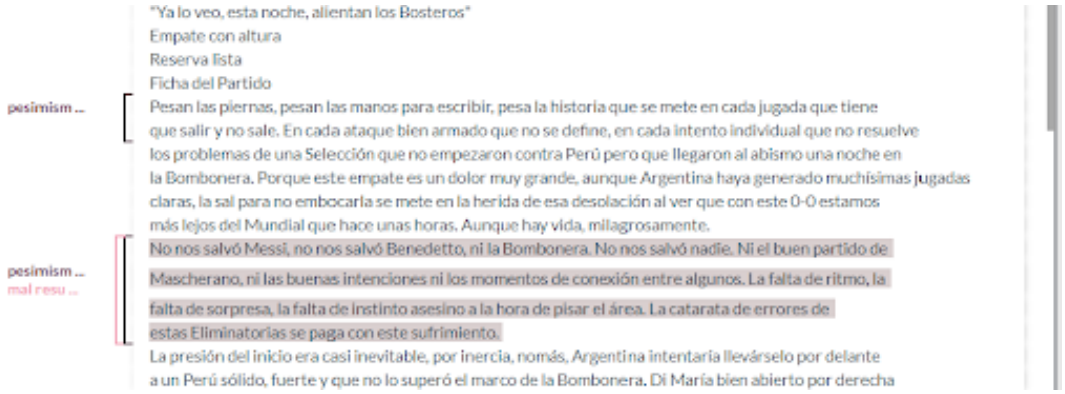



FIGURA 4.6: Ejemplo de una cita

## 4.4. Estructura del texto

La presentación de los textos a analizar es fundamental para una buena experiencia de usuario en una herramienta de las características antes presentadas. Debe ocupar un sector relevante de la pantalla y la interacción con el mismo debe ser tanto fluida como ágil. La estructura arborescente del DOM no es apropiada para trabajar con cadenas de texto. Es difícil identificar dado una palabra, o un rango de palabras, a qué partes del DOM corresponden (*Why Quill*) . Afortunadamente, la solución a dicha característica existe en diversas librerías desarrolladas tanto de forma libre [24] como licenciada. Cada una, con distintas estrategias, permite añadir al sistema un conjunto de funcionalidades que facilitan la exposición del texto como también la posibilidad de editarlo y personalizarlo. Sin embargo, en todos los CAQDAS analizados previamente, y como parte de la herramienta que se quiere desarrollar, existe otra característica que se añade complejidad al despliegue del texto. Como se mencionó anteriormente, las citas corresponden a una porción de texto que tiene asociada ningún (solo tener un memo) o más códigos. (El caso de solo mantener un memo, sería a nivel de visualización igual a tener un solo código por lo que para esta sección se asumen como lo mismo). Para poder visualizar dicha relación se despliega un corchete que delimita el inicio y fin de la cita, además del color del código que representa. Por lo tanto, a cada cita se le añade del lado izquierdo tantos corchetes como códigos tenga asociada como se puede observar en la Figura 4.6.

Lograr mostrar los corchetes de forma contigua al texto que referencian, no es una funcionalidad nativa de las librerías sugeridas antes, y representan un gran desafío para el desarrollo del sistema. Existen varios factores que argumentan la afirmación anterior, a continuación se listan algunos:

- Los corchetes deben mantenerse alineados en todo momento, incluso si el texto de la línea "cae" en consecuencia del tamaño de la pantalla.
- Pueden existir varias citas sobre las mismas oraciones o incluso que coincidan algunas palabras, por lo que los corchetes pueden solaparse. Este comportamiento es aleatorio y depende de la forma en que se codifique el texto.
- Los corchetes aparecen y desaparecen de forma dinámica con el agregado o eliminado de citaciones. Incluso se puede eliminar un código de una cita, sin que esta desaparezca, pero sí, uno de los corchetes que tiene asociada.

[24] quilljs.com

- Cada corchete debe poder reaccionar a eventos en pantalla. Tanto al hacer click o al pasar con el puntero por encima.
- La estructura del texto en pantalla, como los corchetes que la acompañan, se debe persistir de alguna forma para poder reconstruirla cada vez que se ingresa al sistema.

Las problemáticas listadas se han resumido en tres objetivos concretos:

1. Crear una estructura en memoria que permita visualizar el texto de forma amigable y en coordinación con los corchetes que lo acompañan. Además dicha estructura debe soportar eventos de *click* y *mouseover* del navegador.
2. Crear una lógica que permita crear la estructura anterior en tiempo de ejecución y que la actualice cada vez que se elimine o agregue un corchete.
3. Guardar la información mínima indispensable para poder reconstruir la estructura en pantalla en cualquier momento.

### 4.4.1. Objetivo 1

La estructura a crear debe permitir, en primer lugar, disponer el texto en una forma similar a una hoja en un editor clásico. Se crea entonces el concepto de Línea. Este es una abstracción de una línea de texto en su descripción literal, que añade más información. Cada objeto línea, guardará en sus atributos el texto en cuestión y una referencia a todas las citas que refieren en algún punto a esa cadena de caracteres. Mantener la referencia a las citas, permite de forma transitiva tener referencia a los códigos asociados. Por lo tanto, se puede fácilmente crear funciones que utilicen como insumo información de estas tres entidades y tener la certeza que cada instancia está relacionada.

A continuación se crea una estructura tabular, la cual finalmente se traduce en nodos del DOM en los cuales cada fila de una tabla mantiene una relación 1 a 1 con una línea. (En adelante se utilizará el término fila o línea de forma indistinta). Luego cada columna de 1..j-1 hace referencia a un corchete o está en blanco, es decir a un código asociado a una cita asociado a una línea. Afortunadamente, como se dijo antes esta relación está toda embebida en la estructura de línea. De aquí en más a esta referencia de código a cita dentro de una línea se le denominará "porción de corchete". Por lo tanto, se puede decir que las celdas 1..j-1 representan porciones de corchete. Finalmente la columna j contiene el texto a desplegar. El cuadro 4.1 muestra una representación de lo que se describe y la figura 4.7 como se visualiza en pantalla.

| | | | |
|---|---|---|---|
| codigo C - cita 3 - l1 | codigo B - cita 2 - l1 | codigo A - cita 1 -l1 | texto l1 |
| codigo C - cita 3 - l2 | codigo B - cita 2 - l2 | | texto l2 |
| codigo C - cita 3 - l3 | codigo B - cita 2 - l3 | | texto l3 |
| codigo C - cita 3 - l4 | codigo B - cita 2 - l4 | codigo D - cita 4 -l4 | texto l4 |

CUADRO 4.1: Estructura tabular de texto y cita

Sería lógico pensar que esta estructura no es escalable ya que, por cada cita que tenga asociada n códigos que crean 1 x n = n columnas. Pero no todas las filas de cada columna estarán ocupadas, es decir no todas las celdas tienen porciones de corchete, en realidad la mayoría de estas celdas no representarán información alguna.

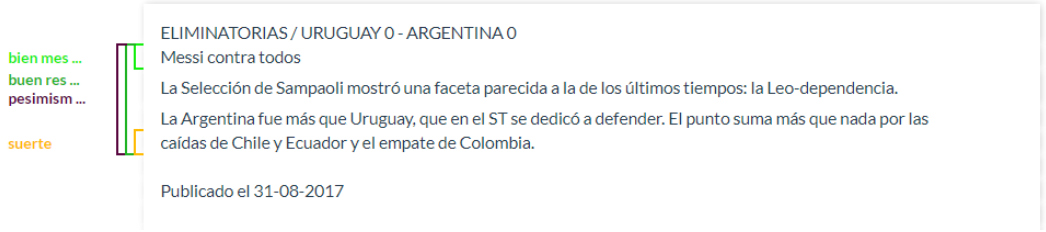



FIGURA 4.7: Representación gráfica del cuadro 4.1

Esto llevaría a que la tabla en el DOM ocupara demasiado espacio inutilizado. Afortunadamente el DOM admite que cada celda de la tabla tenga un ancho arbitrario. De esta forma si la posición (i, j) de tabla es vacía, la celda en cuestión tendrá ancho 0. En cambio una celda ocupada tendrá un ancho determinado y el color que le corresponde según su código. Por lo tanto, sólo aquellas celdas que representan una porción de corchete ocupan espacio en pantalla. Finalmente cada corchete queda determinado por conjunto consecutivo verticalmente de porciones de corchete.

Angular2 a su vez admite reaccionar a eventos sobre cada objeto del DOM. De esta forma, en cada celda de la tabla se podrá agregar lógica que reaccione al evento *click* o *mouseover* logrando así finalmente cumplir con el objetivo primario.

### 4.4.2. Objetivo 2

Luego de comprender cómo estructurar la información para mostrar en pantalla, hay que describir de qué forma calcular las posiciones en la tabla. En esta primer desarrollo del sistema no se permitirá editar los textos importados, por lo que la cantidad de líneas, es decir, filas de la tabla, será fija. Sin embargo la cantidad de columnas, crecerá y disminuirá con la actividad de codificación. Por lo tanto, se debe implementar un algoritmo que permita, dado un conjunto de líneas, determinar qué celdas en la tabla resultante estarán ocupadas y qué información deberá ir allí. Este proceso deberá ejecutarse en, primera instancia, siempre que un documento sea cargado en memoria y luego cada vez que se cree o elimine una cita. Para hacerlo más comprensible se describe un caso con el ejemplo anterior. En el Apéndice D se añade un pseudocódigo del algoritmo.

- La cita 1 al ser creada es la única existente. Por lo tanto la referencia al código A puede alojarse en la primera columna junto al texto y en la única fila que necesita. Como no hay más códigos asociados termina el cálculo par la cita 1.
- Luego la cita 2 se crea, pero como inicia en la misma fila que la cita 1 debe alojarse en la segunda columna, y como ocupa 3 filas las celdas (1, n-2) (2, n-2) (3, n-2)(4, n-2) hacen referencia al código B.
- El caso de la cita 3 es idéntico al anterior, solo que en vez de utilizar la columna n-2, lo hace en la n-3 y refiere al código C.
- La cita 4 encuentra fácilmente su lugar, ya que al comenzar en la fila 4, al evaluar la celda (4, n-1) ésta está libre, y es la única que necesita ocupar relacionándola con el código D.

El ejemplo describe una secuencia de cálculo posible. Aunque existen varias casuísticas más, se puede comprender que siempre el objetivo es determinar la columna que ocupará la cada porción de corchete. Luego cada vez que esta distribución

cambia por eliminación/creacion de citas o la eliminación de un código, la estructura de la tabla debe ser re-calculada. Esto puede ser un cuello de botella en relación a la *performance* del sistema al mantener grandes documentos con muchas citas. Esto podría suponer una necesidad de limitar ciertas cantidades. Este tema se cubre en la sección 4.5.

### 4.4.3. Objetivo 3

Finalmente se debe guardar la información para poder reconstruir esta estructura en pantalla. Como se vio hasta ahora, el texto siempre ocupa la columna n de la tabla. Luego la cantidad de filas es fija por documento. Por otro lado, las columnas de las citas se calculan en tiempo de ejecución. De todo lo anterior se concluye que lo único que debe de guardarse es la fila de inicio y de fin de cada cita y las posiciones de inicio y fin en las respectivas filas. Para esto se agrega a la estructura de las citas cuatro atributos que guardan estos valores.

## 4.5. Limitaciones

En el apartado anterior se describió uno de los aspectos más importantes del sistema. El mismo representa la creación de una estructura de datos la cual ocupará lugar en memoria principal durante la ejecución del sistema. Además implica la realización de diversos cálculos cada vez que haya un cambio en la estructura. Por lo tanto, es imprescindible analizar las necesidades de recursos del el sistema y así determinar ciertas limitantes, siempre con el objetivo de que el usuario tenga un buena experiencia con la herramienta.

Se determinan distintos aspectos que pueden repercutir en el rendimiento. A continuación se examinan estos puntos, se plantean ciertas limitantes del sistema y se explica, donde corresponda, como se llega a estas conclusiones.

### 4.5.1. Tamaño de documentos

El tamaño de documentos es una de las limitantes principales de Qode. Para poder encontrar cierta cota superior, se tomó como primer aproximación el tamaño máximo que podrá ocupar un documento en la base de datos. Como se explicó anteriormente, cada documento (como entidad del sistema) ocupa un documento de MongoDB. Por lo tanto, su tamaño máximo no podrá superar el tamaño estándar que determina la base datos, que es 16MB (*Documents*, MongoDB). Cada documento, contiene el texto del mismo, pero además otra información como son la lista de atributos, referencias a citas, entre otra *metadata* como su nombre o fecha de modificación. De los datos mencionados, aquellos cuyo tamaño es variable son los listados de atributos y de referencias a citas, como también el texto. Teniendo esto en cuenta se plantea determinar cuál sería el tamaño máximo (razonable) alcanzable por un documento en el peor caso.

En primer lugar, se asume el caso en que la cantidad de citas del documento es muy importante, por lo que el largo de la lista de referencias representa un tamaño relevante. Además se asume que la cantidad de atributos también es importante. Para este escenario se plantea un caso que se entiende exagerado. Las listas y el resto de la metadata ocupa 8MB. De esta manera, el texto podría abarcar los restantes 8MB. Si se tiene en cuenta que MongoDB almacena los datos de tipo *string* de la siguiente forma: un número que marca el largo del *string* (se almacena en 4 bytes), el dato almacenado en formato UTF-8, y un *byte* arbitrario que marca el fin del texto. Eso

quiere decir que hay 8.388.608 *bytes* disponibles para el texto. Mediante herramientas para determinar el tamaño en *bytes* de un texto [25] se puede ver que una página normal con líneas largas y cargadas de texto habitualmente ocupa en el entorno de los 3000 *bytes*. Esto tiene sentido ya que en UTF-8 las letras comunes de nuestro abecedario ocupan 1 *byte* (Davis, 1999), y es razonable pensar que una página tenga 30 y pocas líneas cercanas a los 100 caracteres por líneas. Se da que ocupen más de un *byte* las letras con tilde, cedilla u otros símbolos, la mayoría ocupando 2 y habiendo símbolos muy poco habituales en la mayoría de los textos en español que podrían ocupar como máximo 4 *bytes*. Por lo tanto si se busca realizar el cálculo para el peor caso, se puede considerar un texto donde todas las páginas pesen 5000 *bytes*, suponiendo que se usan muchos símbolos que ocupen más espacio y tal vez tenga tamaño de fuente más pequeño, lo que permite más caracteres por página. Aún así, se podrían almacenar textos de hasta 1677 páginas.

El análisis anterior, aunque no totalmente exhaustivo, plantea una situación borde en la cual un documento, y específicamente, su texto, tiene un gran tamaño. Este, queda limitado por la base de datos, pero, este límite, aún no aproxima una cota real al problema. Lo que sucede es que, como se ha descrito antes, la aplicación soporta su lógica en el navegador. El mismo, es un proceso más del sistema operativo, el cual está limitado por la memoria principal. Aquí el análisis se torna más complejo, porque esta limitante varía en cada ordenador. Se podría intentar replicar un análisis teórico del problema, pero se entiende que dicha aproximación depende de muchos factores y por lo tanto se tornaría muy compleja e inexacta. No solo influye cuanto espacio de la memoria principal termina ocupando el proceso del navegador, sino también cuantos otros procesos se ejecutan concurrentemente, como también otros aspectos de *hardware* como la CPU. Por lo tanto se optó por realizar un análisis empírico, tomando fijas las características más comunes de memoria RAM y CPU hoy en día en un ordenador de escritorio.

La metodología utilizada consta de importar documentos cada vez de mayor tamaño y observar cómo repercute en el sistema. Dichas observaciones se realizan sobre el porcentaje de memoria RAM y de CPU que utiliza el proceso del navegador al trabajar con estos archivos.

Teniendo una PC con 8GB de memoria RAM y un procesador I7 se obtuvieron los siguientes resultados:

- **Prueba con documentos de 40 líneas y sin corchetes:** tiempo de carga de documento: 625ms (respuesta del servidor llega a los 550ms). Tiempo de cambio entre documentos: 20ms
- **Prueba con un documento de 40 líneas y con corchetes ocupando menos de 3 columnas:** tiempo de carga de documento: 800ms (respuesta del servidor a los 700 ms)
- **Prueba con varios documentos de 40 líneas y con corchetes ocupando menos de 3 columnas:** tiempo de carga de documento : 820ms , (respuesta del servidor a los 710ms). Tiempo de cambio entre documentos: 60ms
- **Prueba con un documento de 40 líneas y con corchetes ocupando 10 columnas:** tiempo de carga del documento: 900ms (respuesta del servidor llega a los 700 ms). Tiempo de cambio de documento: 75ms

[25] www.mothereff.in

- **Prueba con un documento de 5900 líneas y sin corchetes:** tiempo de carga del documento: 3970ms (la respuesta del servidor llega a los 870ms). Tiempo de cambio de documento: 1900ms
- **Prueba con un documento de 5900 líneas y con corchetes ocupando 3 columnas:** tiempo de carga del documento: 5940ms (la respuesta del servidor llega a los 2570ms). Tiempo de cambio de documento: 3470ms

De las pruebas realizadas se pueden realizar ciertas observaciones.

1. Textos de gran tamaño influyen enormemente en el consumo de memoria RAM. Específicamente, utilizando algunas herramientas de análisis [26] que provee el navegador se puede asegurar que mayormente repercute en los tiempos de renderización.
2. Mantener varios documentos en distintas pestañas no repercute notoriamente en la *performance* del sistema.
3. El agregado de citas al documento tiene un impacto significativo en los tiempos de renderización en documentos grandes.
4. El tiempo que toma calcular la posición del corchete de las citas es bajo.

Analizando las observaciones anteriores y teniendo en cuenta la estructura tabular que se diseñó para mantener el texto y los corchetes de las citas en el DOM, se puede concluir que:

- Grandes volúmenes de texto, generan gran cantidad de líneas en la tabla que mantiene el DOM. Por lo tanto, la cantidad de filas en dicha tabla crece con el tamaño del texto a representar, aumentando el tamaño del HTML.
- El agregado de citas obliga a crecer aún más el tamaño del HTML. Esto sucede ya que cada cita genera al menos una (cada cita puede tener más de un código) columna más de la tabla en el DOM, generando tantas celdas nuevas como líneas existan. Por lo tanto, el agregado de varias citas que se superpongan o con varios códigos asociados, generará varias columnas extra en la tabla que harán crecer el HTML significativamente.
- El tamaño del documento afecta la *performance* del sistema solamente cuando este está siendo visualizado.
- Los tiempos de cálculo de las posiciones de los corchetes que representan a la cita son despreciables ante los tiempos de renderización.

A partir del análisis realizado, se cree correcto afirmar que para asegurar una buena experiencia de usuario, se debe limitar el costo de renderizado del navegador. Para ello es imprescindible limitar el tamaño del HTML generado al desplegar cada documento. Esta limitante queda determinada por el tamaño del texto y la cantidad de porciones de corchetes por línea. Una observación a realizar es que al estar limitada la *performance* por el tamaño del HTML y por la organización tabular del contenido textual del documento, el límite de tamaño del texto no debería ser determinado por el largo de la cadena de caracteres sino por la cantidad de líneas que esta genera. Ya que si la cadena no es larga pero contiene muchos saltos de línea puede generar una tabla muy larga en el HTML. Por lo tanto el umbral que se establece es un largo máximo en función de la cantidad de líneas del documento.

[26] developer.chrome.com/devtools

### 4.5.2. Cantidad de citas por línea

Como se explicó en el punto anterior la cantidad de citas influye en el tiempo de renderización del navegador. Es entonces razonable definir una cantidad máxima de citas por línea. De cualquier manera, hay que tener en cuenta que no son las citas en sí mismas el problema sino las porciones de corchetes que estas definen. En el caso que las citas comparten líneas, los corchetes se solapan debiendo ocupar una columna más de la tabla como se observa en la figura 4.7. Por lo tanto una nueva cita influirá en el tamaño de la tabla, dependiendo de si misma (cantidad de códigos) y las citas que la rodean.

De todo lo anterior se puede concluir que el valor a limitar es la cantidad de columnas de la tabla que representan al documento, es decir las porciones de corchete que determina cada cita. Este número se establece para evitar que la experiencia del usuario se torne poco interactiva en caso de estar visualizando documentos de gran tamaño, ya que para documentos cortos se puede tener una gran cantidad de citas por línea sin que esto afecte los tiempos de renderización. Sin embargo, sí se compromete la visualización del texto ya que a medida que se agregan corchetes el espacio correspondiente al texto comienza a colapsar hacia la derecha de la pantalla.

Este límite, como en el punto anterior se determina empíricamente.

### 4.5.3. Cantidad de citas por documento

El análisis del punto dos ya genera una cota superior para la cantidad de citas de un documento. El valor máximo de citas de un documento se alcanzaría cuando todas las citas tienen asociadas un solo código y cada una sea haga sobre una sola línea. Sea $\lambda$ el valor máximo determinado en el punto 2, la cantidad máxima de citas será igual a la cantidad de líneas del documento multiplicado por $\lambda$.

### 4.5.4. Cantidad de códigos por cita

Es simple deducir que la cantidad de códigos asociados a una cita está acotado por la cantidad de porciones de corchete que puede tener una línea. En el caso en que una cita no comparte líneas con otra, la cantidad máxima de códigos de la cita será $\lambda$.

### 4.5.5. Niveles en sistema de códigos

El sistema de códigos debe de limitarse por el impacto visual de los niveles que genera. El espacio en pantalla que utiliza es relativamente limitado por lo cual se debe acotar.

### 4.5.6. Cantidad de documentos abiertos

No se encuentra un límite técnico para este punto. Simplemente tener en cuenta que cada documento abierto genera una nueva pestaña lo cual puede repercutir visualmente.

### 4.5.7. Cuadro de límites

De los puntos anteriores se define el cuadro 4.2 de límites máximos para las medidas mencionadas. Éstos se determinaron empíricamente realizando diversas pruebas y ajustando los resultados.

| | |
|---|---|
| Largo del documento | 5000 líneas |
| Cantidad de columnas para corchetes por línea | 10 |
| Citas por documento | determinado por orden de citas |
| Niveles en el sistema de códigos | 3 |
| Documentos abiertos | ilimitado |

CUADRO 4.2: Límites sistema

## 4.6. Seguridad

Para la seguridad del sistema se plantean dos elementos a tener en cuenta como son la autenticación y autorización. La autenticación verifica la identidad del usuario permitiendo el acceso al sistema y la autorización comprueba que el usuario tenga los permisos correctos para el acceso al recurso requerido. Ambos conceptos trabajan en conjunto, primero ocurre la autenticación y luego la autorización.

### 4.6.1. Autenticación

Para la autenticación una posibilidad sería el desarrollo de un manejo de usuarios en la propia aplicación, lo cual implica la implementación de funcionalidades como creación de usuarios, cambio/reinicio de contraseña, guardado en una base de datos con seguridad, encriptado de datos para las solicitudes al servidor, restricción de acceso únicamente para usuarios autorizados, entre otras.

En la actualidad, las aplicaciones *web* presentan constantes amenazas y violaciones de datos, en particular se manejan datos que podrían llegar a ser confidenciales y el manejo de los mismos tiende a ser complejo y sensible, por lo cual se requiere una solución lo más fuerte posible. Se decidió entonces que sea un servicio externo el que se encargue de esta complejidad.

En una aplicación tradicional, cuando el usuario realiza un login exitoso una sesión es establecida por el servidor, y esta sesión luego es utilizada para establecer si el usuario se encuentra autenticado o no. En este caso tendremos una api rest con Eve, por lo tanto sin estado, que no almacena sesiones.

Auth0 presenta una solución simple para el manejo de usuarios y su autenticación, la cual se realiza a través de *tokens* que van del servidor al cliente, particularmente JSON *Web Tokens* [27] (JWTs). Entre otras características, con Auth0 se tiene una pantalla de login con soporte para ingreso seguro, ya sea desde un registro nuevo, el inicio de sesión con credenciales o con una cuenta de Google.

JWT es un estándar abierto que define un modo para transmitir de forma segura la información entre las partes como un objeto JSON. Esta información puede ser verificada y es confiable porque está firmada digitalmente utilizando JSON *Web Signature* [28] ( JWS ). La estructura de un JWT consiste en tres partes separadas por puntos (.):

- *Header*: Incluye el tipo del *token*, y el algoritmo de *hash* utilizado. En nuestro caso RS256.
- *Payload*: Contiene las claims, declaraciones sobre una entidad (generalmente, el usuario) y metadatos adicionales.

[27] jwt.io
[28] tools.ietf.org/html/rfc7515

- *Signature* : Usada para verificar que el que envía sea quien dice ser y asegura que el mensaje no fue modificado en el trayecto.

Habitualmente JWT no va encriptado, solo codificado en base64 [29], así que la comunicación podría estar expuesta a ataques de tipo *"Man in the middle"*. Una forma de evitar estos ataques es utilizar https.

### 4.6.2. Autorización

Es un requisito de la aplicación que solo usuarios autenticados puedan acceder a los recursos y a su vez, estos usuarios podrían tener diferentes permisos para la creación, visualización, modificación y borrado de diversos contenidos. Surge entonces la necesidad de usar roles para administrar el acceso de los usuarios a los diferentes recursos del sistema, en este caso, a los proyectos. Los roles permiten aplicar las mismas reglas a un grupo de usuarios. Un usuario puede tener distintos roles según al recurso al que se referencia. Por ejemplo, un rol de solo lectura en un proyecto 'Y', no permitirá al usuario la creación de nuevas citas. La implementación de permisos se realiza tanto del lado cliente *web* como en el servidor. El uso de *tokens*, es también la forma con la que se realiza la validación de permisos en el servidor.

### 4.6.3. Funcionamiento

El *back-end* no necesita mantener un registro de los *tokens*. Cada uno es compacto, lo cual permite que sea pasado como parte de la cabecera HTTP de la petición de una forma rápida; y es autocontenido dado que contiene todos los datos necesarios para comprobar su validez, así como información del usuario.

El token es enviado bajo la clave 'Authorization' de la siguiente forma:

*"Bearer- token*

Estas características permiten la escalabilidad inmediata ya que las peticiones no dependen unas de otras. De esta forma se podrían en el futuro de la aplicación realizar en diferentes servidores de forma independiente.

Se describe a continuación el flujo normal de acceso a la aplicación y las implementaciones realizadas en cada caso. El usuario accede a una pantalla de login, donde procede a registrarse, ingresar sus credenciales de un registro previo o acceder con su cuenta de Google. En el registro se requiere el ingreso de una contraseña que respete ciertas características para hacer más segura a la misma. Al iniciar sesión en la aplicación, una vez Auth0 verifica sus credenciales, retorna un *token* como respuesta. El usuario persiste en el *localstorage* del navegador el *token* obtenido que debe enviar en cada petición posterior. En cada una de estas el servidor debe realizar una comprobación de su validez, si este es correcto podrá acceder a los recursos solicitados, y si el mismo es o se torna inválido, cualquier operación que el usuario realice para acceder al servidor será denegada, resultando en que el mismo deba realizar nuevamente el proceso de inicio de sesión. El *token* expirará después de un determinado tiempo, lo cual añade una capa extra de seguridad.

Para validar dicho JWT, tanto el servidor como el cliente *web* debe conocer: el id de cliente de Auth0, y la clave secreta correspondiente. Estos datos son obtenidos en la cuenta de Auth0 que fue creada.

Además, del lado del servidor se utiliza la información obtenida a través de la decodificación del *token* para identificar al usuario y de esta forma restringir el acceso a los recursos según los permisos que el mismo tenga definidos. Esto se realizó

[29] www.base64decode.org

con la implementación de una regla [30] en Auth0 que agrega como *claim* en el *payload* del *accessToken* [31] un nuevo campo 'email', con valor *e-mail* del usuario. Dicha regla es una función escrita en JavaScript que se ejecuta en Auth0 como parte de la transacción cada vez que un usuario se autentica en la aplicación, y antes de la autorización.

Para el acceso a cada recurso en la API, se tiene entonces además de la validación del *token*, la decodificación y la obtención del *e-mail* que se agregó con la regla. De esta forma se realiza la identificación del usuario en la API de una forma segura. El *e-mail* se utiliza para:

- La asignación del *owner* del proyecto durante su creación.
- acceder a la pantalla de mis proyectos y solicitar los proyectos a la API, se retornan únicamente los que tienen como creador o colaborador dicho *e-mail*.
- La validación del rol del usuario dentro de un proyecto en cada acceso a sus recursos. Este acceso está restringido a nivel de rol y proyecto. En cada proyecto el usuario tiene un rol definido, a partir del mismo se determina qué acciones puede realizar en los recursos que están asociados a dicho proyecto.

Para su desarrollo se utilizaron los Event Hooks [32] de Eve previo a las operaciones de *get* , *post* , *update* ó *delete*. En el *front-end* de la aplicación se implementaron filtros sobre la vista que ocultan o bloquean algunas de las funcionalidades provistas. Al igual que en el servidor, los permisos dependen del rol del usuario dentro del proyecto accedido.

## 4.7. Entornos de trabajo

En todo proyecto donde se realizan liberaciones progresivas de prototipos de versiones estables de producción del software desarrollado es imprescindible realizar la separación del trabajo en entornos. Esto brinda independencia entre el medio en el que se ejecuta la versión de testing, que debe tener un funcionamiento similar al que tendría en producción, con el medio en el que se desarrolla. Al desarrollar se pueden realizar cambios que comprometen parcialmente funcionalidades ya incluídas en prototipos anteriores considerados versiones estables, por esta razón los entornos deben estar separados.

El funcionamiento en ambos entornos debe ser similar, pero los servicios que se consumen como Auth0 y la base de datos deben ser accedidos en distintas direcciones y con distintas credenciales, así como también la API queda disponible en un dominio diferente. Todos estos datos se modifican en dos archivos de configuración, uno para *front-end* y otro *back-end*, como se puede ver en detalle en el Apéndice E sobre la instalación de la herramienta.

Durante el proceso de desarrollo de Qode se dispone de dos tipos de entornos, descritos a continuación.

#### Entorno de desarrollo

El ambiente de desarrollo no es uno específicamente, sino que hay un ambiente de desarrollo distinto para cada miembro del equipo. Estos tienen las mismas características con la diferencia de que no usan el mismo sistema operativo.

[30] auth0.com/docs/rules/current
[31] auth0.com/docs/tokens/access-token
[32] python-eve.org/features.html

Dichas características son: Se ejecutan dos aplicaciones separadas para la API y para el *front-end*. La API se ejecuta en un servidor provisto por la librería *gevent* en Python, el mismo ejecuta la aplicación en el puerto 5000 localmente. El *front-end* se ejecuta mediante la herramienta Angular-CLI [33] que provee las funcionalidades necesarias para implementar, testear, desplegar y empaquetar proyectos Angular, mediante el comando *ng serve* se realiza el despliegue de la aplicación de *front-end* en el puerto 4200. Para la persistencia de datos se usa una instalación local de MongoDB y para la autenticación de usuarios se usa el servicio Auth0 con las configuraciones correspondientes a un servidor local.

**Entorno de *testing***

El ambiente de *testing* tiene características similares a uno de producción y se usa para probar versiones de prototipos de la herramienta. A lo largo del proceso de desarrollo se realizan liberaciones de versiones marcadas por conjuntos de funcionalidades como hitos.

Para disponer de un servidor similar a un entorno de producción donde la aplicación debe tener una disponibilidad muy alta se usa una plataforma de servidor *web* en la nube, la misma se llama Heroku [34] y provee planes gratuitos para aplicaciones de pequeño porte y con una cantidad de 500 horas mensuales de ejecución, con la particularidad de que cuando ningún usuario la está accediendo ésta permanece inactiva, pausando el conteo de horas. En esta plataforma se ejecutan dos aplicaciones por separado, una encargada del despliegue del *front-end* y la otra corriendo la API como servidor. Para la base de datos se utiliza un servicio provisto por terceros, disponible en Heroku como *add-on*, el mismo se llama mlab[35] y provee de una base de datos con estructura estilo MongoDB por lo cual es compatible con la aplicación. La API se comunica con este servicio para persistir los datos. Para el servicio de autenticación se usa Auth0 al igual que para el ambiente de desarrollo, pero con una cuenta distinta para mantener los usuarios de ambos entornos de forma separada, para esta cuenta se configura con las direcciones correspondientes al entorno de Heroku. En el Apéndice E se agrega un manual para poder realizar el *deploy* del sistema en Heroku.

En el entorno de *testing* se usa una herramienta llamada "heroku-buildpack-static"[36] que sirve para el despliegue de páginas estáticas o *single page*, se usa para desplegar los archivos generados en el empaquetado de la aplicación de *front-end* mediante Angular-CLI. Se usa esta herramienta para usar el servicio de *proxy0* que brinda y evitar que los exploradores bloqueen los llamados a la API por la política del mismo origen. Esta política es una medida de seguridad llevada a cabo por los navegadores que no permiten que desde un script que se ejecuta en el cliente se pueda cargar contenidos o realizar llamados a orígenes (se usa como sinónimo de dominio) distintos al de la página que se está ejecutando. Para la comunicación entre la API y el *front-end* no se tienen problemas porque mediante el protocolo CORS (Cross Origin Resource Sharing) que implica agregar ciertos cabezales que declaran a quien se le da permiso a comunicarse entre orígenes distintos se hace que la API otorgue dichos permisos a la aplicación de *front-end*. El problema se da al querer comunicarse desde el *front-end* con Auth0 ya que este servicio no tiene una manera

[33] cli.angular.io
[34] heroku.com
[35] mlab.com/company
[36] github.com/heroku/heroku-buildpack-static

sencilla de resolver este problema. Se usa el *proxy* para que los llamados que se realizan a Auth0 se hagan a una dirección interna del dominio de la aplicación pero este se encargue de hacer el llamado al verdadero dominio de Auth0.

Otro beneficio de usar esta herramienta para el despliegue de la aplicación de *front-end* es que se puede configurar la misma para que sólo se pueda acceder mediante https. Aprovechando que Heroku brinda tanto el protocolo http como https y haciendo uso de la posibilidad de obligar el acceso mediante https se brinda este tipo de seguridad en el entorno de *testing*, algo que se debería realizar también en cualquier entorno de producción donde se quiera dejar disponible la aplicación de modo de brindar la mayor protección posible a los datos del usuario.

# Capítulo 5

# Conclusiones y trabajo futuro

En este capítulo, se detallan primero las características de Qode habiendo finalizado el período de desarrollo bajo la actividad de Proyecto de grado. Luego se sugieren los próximos pasos a dar en posibles trabajos futuros que se consideran de mayor utilidad para agregar a la herramienta. Finalmente se realizan las reflexiones finales por parte del grupo sobre el proceso de investigación y desarrollo, y el valor del resultado obtenido.

## 5.1. Herramienta lograda

En esta sección se muestra hasta qué punto del desarrollo planteado se llegó. Al comienzo de la etapa de implementación, se propuso un desarrollo en etapas para garantizar que el resultado obtenido fuera un prototipo estable de la herramienta, más allá de si el tiempo alcanzase o no para todo el desarrollo planeado.

Dentro del tiempo acotado por la actividad de proyecto de grado se llegó completar parcialmente la etapa 2. De la etapa 3 no se llegó a incluir ninguna funcionalidad.

El ingreso de los datos a Qode estaba originalmente pensado como un módulo independiente asignado a otro proyecto de grado, de todas formas la versión actual de Qode cuenta con un servicio REST para cargar datos de forma directa al sistema.

A continuación se visualizan nuevamente los cuadros del capítulo 3, agregandose una columna de "Logrado"donde se marca si lo propuesto en dicho capítulo fue desarrollado o no.

**Funcionalidades básicas**

| N | Nombre | Incluido | Etapa | Logrado |
|---|---|---|---|---|
| 1 | Codificación | SI | 1 | SI |
| 2 | Variables/Atributos | SI | 2 | SI |
| 3 | Memos/Anotaciones | SI | 1 | SI |
| 4 | Búsquedas de texto | SI | 2 | Parcial |
| 5 | Búsquedas de código | SI | 2 | SI |
| 6 | Modelos gráficos | SI | 2 | Parcial |
| 7 | Analisis dimensional | SI | 3 | NO |
| 8 | Hipertexto | SI | 3 | NO |
| 9 | Reportes | SI | 3 | NO |

CUADRO 5.1: Funcionalidades básicas.

**Funcionalidades avanzadas**

| N | Nombre | Incluido | Etapa | Logrado |
|---|---|---|---|---|
| 1 | Sugerencias | NO | - | - |
| 2 | Búsquedas semánticas | NO | - | - |
| 3 | Auto codificación | NO | - | - |
| 4 | Frecuencia de palabras | SI | 3 | Parcial |
| 5 | Trabajo colaborativo | Parcial | 1 | Parcial |
| 6 | Atajos de teclado | SI | 2 | Parcial |

CUADRO 5.2: Funcionalidades avanzadas.

**Requermientos cliente**

| N | Nombre | Incluido | Etapa | Logrado |
|---|---|---|---|---|
| 1 | Carga directa de datos | SI | 1 | SI |

CUADRO 5.3: Requerimientos cliente.

Sobre las funcionalidades que quedaron desarrolladas parcialmente en etapa 1 y 2, se detallan las distintas razones de cada una.

1. **Búsquedas de texto:** Esta funcionalidad implica la búsqueda de una cadena de caracteres ingresada por el usuario en todos los documentos abiertos, permitiendo el avance y el retroceso entre los resultados encontrados, mostrando en pantalla y pintando la ocurrencia actual, su índice y la cantidad total de coincidencias encontradas. La implementación de dicha funcionalidad representó dificultades por las características de la estructura del documento y el procesamiento que se realiza en la importación de los distintos formatos soportados por la herramienta. La funcionalidad fue desarrollada pero presenta algunos errores principalmente de rangos en la selección y se decidió por lo tanto no incluirla en la versión estable.

2. **Modelos gráficos:** Si bien se incluyeron funcionalidades gráficas que dan una idea de la interrelación entre códigos y la ocurrencia de códigos en los distintos documentos, el potencial en esta área es tan grande que hay un largo camino por recorrer antes de poder considerar que está completamente implementada.

3. **Trabajo colaborativo:** Qode permite a los usuarios compartir proyectos con manejo de roles para poder controlar los permisos de edición, esta funcionalidad no es menor pero para brindar la capacidad completa de trabajo colaborativo falta implementar alguna forma sincronización de los recursos en tiempo real. En la próxima sección se explica como se puede llevar a cabo.

4. **Atajos de teclado:** Se incluyeron atajos de teclado definidos para las tareas más importantes para la codificación. Se marca esta funcionalidad como lograda parcialmente por el hecho de que no se incluyó la posibilidad de que el usuario configure sus propios atajos de teclado. Si bien los atajos actuales agregan agilidad a la codificación aún se puede ir más lejos dando la funcionalidad de su personalización.

Se entiende que a pesar de haber alcanzado el desarrollo de la mayoría de las funcionalidades que fueron planteadas al inicio del proyecto, es claro que todas ellas pueden ser mejoradas. La siguiente sección se centra en analizar el resto de las funcionalidades que no fueron incluidas en el alcance pero que potenciarán de gran manera a Qode.

## 5.2. Trabajo futuro

Como se ve en la sección anterior Qode provee varias funcionalidades muy útiles a la hora del análisis cualitativo que cubren las tareas necesarias para cubrir esa etapa de la investigación. Además, como se explicará más adelante en la sección 5.3, el desarrollo logrado cumple con los objetivos planteados al inicio del proyecto, es de utilidad para el cliente, siendo una herramienta libre y de fácil aprendizaje. Sin embargo, en comparación con los software vistos en el estudio del estado del arte, Qode no deja de ser una herramienta con funcionalidades básicas.

Esto muestra que hay un gran potencial de desarrollo de funcionalidades futuras. A continuación se describen algunas ideas que podrían darle a la aplicación un salto de calidad importante.

#### Trabajo colaborativo

Qode da soporte al trabajo colaborativo, se permite tener proyectos multiusuario, teniendo distintos niveles de permisos de edición. A su vez, al ser una herramienta web, si se la aloja en un servidor se habilita el acceso desde distintas computadoras. Sin embargo, esta funcionalidad no es completa ya que si el trabajo colaborativo se realiza en simultánea un usuario no ve los cambios que está realizando el otro si no refresca la página para volver a cargar los datos desde el servidor. Por las características de las metodologías para análisis cualitativo de datos, se entiende que es poco habitual que dos investigadores codifiquen un mismo documento en simultánea, es por esto que el desarrollo de la funcionalidad de trabajo colaborativo "en vivo" no fue considerada prioritaria por el equipo de desarrollo. De todas formas si se quiere que se difunda el uso de la herramienta lo ideal es no sesgar el universo de metodologías que se pueden usar con ella, realizar esta mejora le daría a los usuarios más libertades, sin preocuparse de si dos o más usuarios realizan acciones simultáneamente sobre el proyecto.

Por las características arquitectónicas de Qode, donde gran parte de la lógica se encuentra en el *front-end* y se realizan llamadas a la API de *back-end* de forma de obtener los datos, hay muy pocos momentos en los que se refrescan los datos del proyecto cargándose la información desde el servidor. Los datos correspondientes a los recursos internos del proyecto como documentos, códigos, citas y memos se cargan al acceder al *workspace*, pero luego las modificaciones que el usuario hace sólo realizan llamados para persistir los datos pero no extraer información del servidor. Por lo tanto, la manera correcta de lograr el objetivo es mediante WebSockets, ésta es una tecnología que permite comunicación bidireccional en directo entre el navegador y el servidor mediante un socket TCP. Esta es la manera más adecuada, ya que si se optara por cargar los datos solamente ante ciertas acciones del usuario y éste está dedicado exclusivamente a leer un documento, sin generar un evento ante el cual el programa reaccione, no se actualizaría en ningún momento la información. De esta forma nunca se enteraría de las acciones que otro usuario realiza en simultánea sobre el documento.

Es bastante normal en las metodologías cualitativas que un investigador esté un cierto tiempo dedicado solamente a la lectura exhaustiva del documento previo a comenzar la codificación, por lo que largos períodos sin interacción con el servidor no son extraños. Otra posibilidad es realizar llamados al servidor cada ciertos intervalos de tiempo para actualizar los datos de los recursos, pero esto es poco performante, ya que si nadie más está modificando datos del proyecto al mismo tiempo que el usuario trabaja no tiene sentido estar todo el tiempo realizando llamadas al servidor y podría empeorar la experiencia de uso.

**Visualización de documentos**

**Formato**

La forma en que se visualiza el documento sobre el cual se está trabajando permite una lectura cómoda y cumple con lo imprescindible para una herramienta de este tipo, poder codificar e interactuar con todas las funcionalidades del workspace sin necesidad de cerrar el documento de forma de agilizar el análisis. Sin embargo, el formato en que se despliega es como texto plano, esto implica que la forma como se muestra el documento en Qode pueda diferir del archivo original en aspectos como tamaños, estilos y colores de letras, así como también en la separación entre las líneas, las sangrías y los márgenes del texto. Además si el archivo que se importa contiene objetos que no sean texto, como imágenes, tablas o campos de texto especiales( como por ejemplo ecuaciones), los mismos se pierden al cargar el documento en la aplicación. En su etapa actual la herramienta se planteó como un ayuda para el análisis de texto, dejando de lado otro tipo de recursos como imágenes, audio y video, por lo que todas estas características no influyen en dicha funcionalidad.

Sin embargo, está claro que cuánto más completo y similar a un editor sea el componente donde se despliega el texto, mejor. Lo ideal para brindarle al usuario la mejor experiencia posible es que el documento se muestre con los mismos formatos que el archivo original, y que se puedan ver los objetos embebidos ( imágenes, tablas, ecuaciones, etc). Teniendo un visor de texto con esas características también se puede agregar la capacidad de editar el contenido del texto, de forma que si el usuario quiere realizar modificaciones pueda hacerlo sin tener que usar un editor externo.

Estas mejoras no son sencillas, ya que implican ciertos problemas en la forma de almacenar el texto respetando los formatos del mismo y sin que se genere un gran impacto en el tamaño de almacenamiento. Otra problemática es la traducción del cuerpo del archivo a importar ya que dependiendo del tipo de archivo hay que realizar distintas conversiones para que al desplegarlo en la página usando código HTML mantenga las características originales.

**Renderización**

Como ya se mencionó en la de limitaciones del Capítulo 4, un problema que tiene la herramienta es que al visualizar un documento muy grande se carga todo el documento y el código HTML generado es muy largo. Esto implica un gran aumento en el consumo de memoria y en los tiempos de respuesta de la herramienta. Si bien se recomienda a los usuarios usar documentos no demasiado extensos, como se dijo anteriormente, lo ideal es darle las mayores libertades posibles. Una manera de resolver este problema es lograr que al seleccionar el documento no se despliegue todo el texto a la vez, sino que se despliegue un número acotado de páginas, y que el conjunto de páginas desplegadas se vaya modificando conforme el usuario avanza

en el documento. Esta modificación no es sencilla, ya que repercute en varias de las funcionalidades actuales por ejemplo en la forma en que se manipula una citas en caso de que al generarla esta ocupe una cantidad de páginas mayor a la de un conjunto de páginas desplegable, lo cual se puede considerar una situación rara pero no imposible.

**Automatización en la codificación**

Actualmente Qode permite varias maneras de crear códigos, pudiendo hacerlo directamente al momento de codificar un segmento de texto, o previamente con maneras diferentes, una de creado rápido poniendo sólo el nombre y otra permitiendo editar todos los datos del código. También se permite codificar el texto de dos formas distintas, creando una cita y agregándole los códigos pertinentes, o usando la modalidad de crear una cita y que automáticamente se agreguen a esta todos los códigos activados por el usuario.

Sin embargo, por el rol central que tiene la codificación dentro de la metodología, toda facilidad que se le pueda dar al usuario para agilizar este proceso e incluso hacerlo más potente, es bienvenida. Con este fin se plantean dos ramas de desarrollo, una muy sencilla que agrega agilidad al proceso, y otra mucho más compleja pero que además puede agregar gran valor al proceso de análisis.

**Atajos de teclado**

Si bien este tipo de atajos se agregaron para las funcionalidades más útiles de la codificación, podrían servir para cualquier otra que provee la herramienta, como podría ser el acceso a cualquiera de las opciones provistas en el cabezal del workspace. Además, para la tarea de codificación se propone que el usuario pueda tener atajos de teclado asociados a cada código, de forma que al seleccionar un segmento de texto, con sólo usar el atajo de teclado correspondiente a un código puntual, ya crea la cita codificada forma automática. El largo de la lista de códigos asociados a un proyecto es variable, por lo que tener atajos definidos por la herramienta para los códigos no es posible.

Es por esto que la mejor opción es dejar que el usuario tenga la posibilidad de realizar esta configuración para cada uno de sus códigos de forma sencilla, sin necesidad de hacerlo para todos los códigos. Esta funcionalidad le permite tener una experiencia mucho más ágil y personalizada de codificación.

**Automatización mediante PLN y/o ML**

Este tema fue planteado dentro de las posibles funcionalidades a la hora de estudiar el alcance del proyecto pero se descartó rápidamente por su complejidad, priorizando lograr una versión estable que brinde al usuario el flujo completo básico que requiere el análisis cualitativo de texto. Distinta es la situación actual, luego de haber desarrollado una herramienta que cumple con todos los requisitos más elementales se puede poner en discusión funcionalidades como estas.

Como ya se fue discutido al momento de realizar el trabajo de priorización de las funcionalidades a desarrollar, la codificación automática no es para nada sencilla. La diferencia de significados que puede tener una palabra en función de la temática del texto, de la investigación, del registro lingüístico y contexto del documento, agregado a la variabilidad en la granularidad que puede tener una cita (puede ser desde una palabra hasta varios párrafos o el texto entero) hacen que la codificación completamente automática sea además de muy compleja, peligrosa. Se hace esta afirmación

porque si el porcentaje de aciertos de la codificación automática respecto al juicio del usuario no es extremadamente alto, se puede dar que esta ayuda termine siendo una complicación para el usuario que tendría que estar prestando especial atención y corrigiendo errores en el proceso de forma habitual. Se propone no descartar la automatización por estos medios, mas no usarlo para acciones tan drásticas como codificar todo un documento automáticamente, sí hacer uso de estos métodos para ayudar al usuario en el proceso mediante sugerencias.

Algunas funcionalidades interesantes y menos complejas que la codificación automática que podrían desarrollarse:

- **Sugerencia de código existente para segmento de texto seleccionado:** esta funcionalidad tiene una dificultad menor ya que como el usuario seleccionó el texto a codificar se conoce la granularidad de la cita, y el texto a clasificar es más acotado. Se pueden usar técnicas de aprendizaje automático y procesamiento del lenguaje natural, si se usa el primero, seguramente suceda que para las primeras codificaciones la sugerencias no sean tan acertadas, pero a medida que el usuario codifique cada vez más segmentos de texto estas cada vez coinciden más con sus criterios.

- **Sugerencia de nuevo código para segmento de texto seleccionado:** esta funcionalidad se puede realizar mediante procesamiento del lenguaje natural y con apoyo de ontologías que permitan asociar palabras relacionadas a la temática. Una dificultad que se puede dar es que es muy difícil tener una ontología que cubra todas las posibles temáticas de estudio para la investigación, una posibilidad sería darle al usuario la capacidad de agregar ontologías que él considere adecuada para la ayuda en este proceso.

- **Sugerencia de creación de código nuevo:**En este punto el uso de ontologías parece casi imprescindible, siendo el procesamiento del lenguaje también útil. La idea es que la herramienta sugiera posibles códigos nuevos en función de los ya creados por el usuario. Si bien esta funcionalidad no agiliza la tarea de codificación, puede agilizar la creación de códigos y sobre todo puede ser de ayuda para el usuario sugiriendo conceptos que tal vez no se le hubieran ocurrido.

Si se observa el Capítulo 2 , se puede notar que las herramientas analizadas presentan gran cantidad de funcionalidades además de las que presenta Qode. Los CAQDAS por definición son herramientas para asistir al proceso de análisis cualitativo de datos. Esto implica que si bien, por ser herramientas de asistencia no deberían realizar el análisis completo, mientras no brinden la totalidad de acciones necesarias para la variedad de metodologías de análisis cualitativo, siempre habrá alguna funcionalidad para agregar.

Qode provee mecanismos para extraer relaciones entre códigos y documentos mediante las tablas y gráficas de ocurrencias y coocurrencias de códigos por documentos, así como recuperación de citas codificadas, las cuales permiten realizar el análisis de los datos pero es un conjunto acotado de funcionalidades en comparación con las herramientas líderes del mercado. Si bien el objetivo del proyecto en esta instancia no es brindar las mismas funcionalidades que dichas herramientas, estas son un buen indicador de cuáles son más útiles y por lo tanto prioritarias para que el usuario pueda tener la mejor experiencia de obtención de resultados a partir del análisis realizado. Algunas ideas sobre este tipo de funcionalidades ya fueron

planteadas al analizar las funcionalidades básicas y avanzadas que puede tener un CAQDAS en el Capítulo 3.

Además de las funcionalidades ya mencionadas, las cuales aplican directamente al objetivo de Qode que es realizar análisis cualitativo exclusivamente de texto escrito, el trabajo futuro podría implicar expandir el universo cubierto por la herramienta, como podría ser permitir análisis de otro tipo de fuentes como archivos multimedia, o dar ayudas al usuario en la obtención de datos mediante herramientas de transcripción o de toma de notas.

Esto es una clara muestra de lo amplio que es el espectro de acciones útiles a la hora de la investigación cualitativa y por lo tanto del potencial de crecimiento que tiene Qode. Este espectro de funcionalidades excede ampliamente lo que el grupo de desarrollo era capaz de lograr en el tiempo acotado por el proyecto de grado. Por esa razón es que entre los objetivos principales se planteó lograr una herramienta con las funcionalidades básicas pero que sea escalable y de código abierto para permitir que el desarrollo de la herramienta pueda seguir fuera del ámbito de este proyecto de grado.

## 5.3. Reflexiones Finales

Qode es el resultado de un largo proceso, partiendo de un nulo conocimiento del área y muchas dudas con respecto al objetivo y alcance de la herramienta a desarrollar; se logra una aplicación con gran potencial que cumple con todo lo básico y más. A continuación se comentan algunas partes muy importantes del proceso, pero no visibles de forma directa en el resultado final.

Se dedicó una buena parte del tiempo del proyecto al adentramiento en el área de la investigación cualitativa, particularmente en su etapa de análisis; a la comparación de las herramientas existentes y la extracción de las funcionalidades comunes y más importantes, para lograr una aplicación que cumpla con las expectativas de cualquier usuario con o sin conocimiento del área. Destacamos la guía inicial dada por Alén y Luis Pablo, que fueron de gran ayuda para la orientación y el planteo de los objetivos.

Al momento de diseñar la aplicación, se nos planteó el dilema de la elección de las tecnologías a utilizar para el desarrollo, decidiendo utilizar unas con las que no se tenía experiencia. Esto nos planteó un gran desafío en este aspecto, con una alta curva de aprendizaje, que nos llevó a cometer errores en el proceso de los cuales hubo que aprender sobre la marcha.

Particularmente, la inclusión de una tecnología de terceros como es Auth0, tuvo sus grandes ventajas, permitió ahorrar mucho tiempo y pienso en la implementación del manejo de usuarios y seguridad, pero a su vez, agregó dificultades a la hora de desplegar la aplicación.

Los tiempos detallados, si bien algunos fueron mayores de los previstos, forman parte del proceso habitual de aprendizaje que conlleva un proyecto en un área desconocida, además se consideran esenciales para lograr una herramienta consistente con los objetivos planteados.

Nuestra evaluación sobre la realización de los objetivos planteados es positiva. En primer lugar, Qode se planteó como una herramienta de propósito general para abarcar no solo las necesidades de la cliente sino también las de cualquier usuario que se interese por utilizarla. Se brindan funcionalidades que abarcan toda la fase de análisis. Este ciclo comienza con la carga de datos, mediante importación de documentos, de códigos usados en proyectos anteriores y con la capacidad de cargar estas

entidades de forma directa mediante la API, sin necesidad de usar el componente de *front-end*(único requerimiento específico de la cliente). Luego las funcionalidades de codificación y anotación del texto, así como carga de atributos a los documentos permiten la identificación de conceptos para posteriormente usar herramientas de recuperación de segmentos codificados y cálculo de ocurrencias y coocurrencias de códigos que permitan refinamiento del análisis y obtención de resultados.

En segundo lugar, la herramienta desarrollada cumple con el requisito de ser *software* libre. Este punto no es menor si se tiene en cuenta que las principales herramientas del área son de *software* propietario y con licencias muy caras. Este es un punto muy fuerte a futuro considerando que el desarrollo inicial de Qode consistió en un grupo de sólo 3 estudiantes, con plazo acotado y con las dificultades ya planteadas, lo cual marca una gran diferencia con respecto a las herramientas más completas que son realizadas por empresas con grupos más grandes de desarrolladores y dedicados a los CAQDAS. Por ser el código libre, el potencial equipo de desarrollo que se puede generar a futuro es de tamaño ilimitado.

El tercer objetivo que se planteó para Qode es que sea de uso intuitivo y fácil aprendizaje. Creemos que este punto se logró, brindando una interfaz de usuario muy cómoda, con buena visualización del texto a analizar y con acceso intuitivo a las funcionalidades. Además se crearon videotutoriales y un manual de uso para ayudar a los usuarios a la hora de hacer sus primeras armas con la herramienta.

Además de haber alcanzado los objetivos planteado, hay una característica en la que Qode es mejor que las herramientas vistas en el Capítulo 2, esta es su estructura web. Un problema visto en los CAQDAS analizados son las diferencias de características entre sus versiones para distintos sistemas operativos (en los casos que soportan más de uno), en nuestra herramienta esto no sucede ya que si un usuario tiene acceso a un web browser va a acceder a las mismas funcionalidades sin importar el sistema operativo que use. Otro beneficio de ser una herramienta web es que el usuario puede acceder desde cualquier lugar sin previa instalación, a diferencia de los softwares mencionados que no sólo requieren instalar el programa y a veces también algún paquete agregado, sino que además limitan en sus licencias las cantidad de dispositivos permitidos para la instalación.

Tenemos esperanza de que la herramienta lograda genere arraigo en la comunidad de software libre y en investigadores cualitativos que estén introduciéndose al uso de CAQDAS. De esta forma, creemos que se puede lograr el cometido de Qode, mismo cometido que tenía el grupo académico que impulsó LibreQDA. Con el ideal de brindar libertad a los investigadores a la hora de elegir qué herramienta de análisis cualitativo de datos prefieren sin tener que caer en manos de grandes empresas, las cuáles eligen qué acciones se pueden realizar y además fijan precios prohibitivos para muchos usuarios. Qode es software libre y por ende brinda tanto libertad de uso como de modificación, haciendo que la comunidad de usuarios sean los verdaderos dueños de la herramienta.

# Referencias Bibliográficas

*5 Best Javascript Frameworks In 2017* (2017). Recuperado el 19 de Noviembre, 2017, de Da14: https://da-14.com/blog/5-best-javascript-frameworks-2017.

Alarcón Venegas, Jocelyn, Joselin Carrasco Quijada y Ana Pérez Ponce (2012). *Breve historia de la investigación cualitativa*. Recuperado el 15 de Setiembre, 2017, de Pedagogía. Gral. Básica. Universidad del Bío-Bío: http://licentiare.blogspot.com.uy/2012/11/breve-historia-de-la-investigacion.html.

Baquero, José María (2016). *Bases de Datos No Relacionales (NoSQL): cuándo, cómo y para qué usarlas*. Recuperado el 8 de Octubre, 2017, de Silicon: https://www.silicon.es/bases-datos-no-relacionales-nosql-cuando-usarlas-2324948.

Carvajal Llamas, Diógenes (2001). «Herramientas informáticas para el análisis cualitativo». En: *Nómadas* 14. Recuperado el 29 de Setiembre, 2017, de Universidad Central de Colombia: https://www.ucentral.edu.co/images/editorial/nomadas/docs/nomadas_14_18_herramientas.PDF.

Castro Romero, Alexander, Juan Sebastián González Sanabria y Mauro Callejas Cuervo (2012). «Utilidad y funcionamiento de las bases de datos NoSQL». En: *Facultad de Ingeniería* 21.33. Universidad Pedagógica y Tecnológica de Colombia, págs. 21-32.

Davis, Mark (1999). Recuperado el 24 de Marzo, 2018, de International Components for Unicode: http://www.icu-project.org/docs/papers/forms_of_unicode/.

*Diferencias entre software libre, de código abierto y gratuito* (2018). Recuperado el 26 de Abril, 2018, de Somos Libres: http://www.somoslibres.org/modules.php?name=News&file=article&sid=7550.

*Diferencias entre Software Libre y Open Source* (2014). Recuperado el 27 de Abril, 2018, de Hipertextual: https://hipertextual.com/archivo/2014/05/diferencias-software-libre-y-open-source/.

*Documents*. Recuperado el 15 de Febrero, 2018, de MongoDB: https://docs.mongodb.com/manual/core/document/.

Gibbs, Graham R., Celia Taylor y Ann Lewins (2005). *Software tools*. Recuperado el 17 de Octubre, 2017, de Universidad de Huddersfield y Universidad de Surrey: http://onlineqda.hud.ac.uk/Intro_CAQDAS/software_tools.php.

Glaser, Barney G. y Anselm L. Strauss (1967). *The Discovery of Grounded Theory*. Aldine Transaction. ISBN: 9780202302607.

Hernández Pina, F., M.P. García-Sanz y J.J Maquilón. *Metodologías de la investigación en educación*. Recuperado el 12 de Dicembre, 2017 de Unidad de innovación, Universidad de Murcia: http://ocw.um.es/cc.-sociales/metodologias-de-la-investigacion-en-educacion/material-de-clase-1/t1.5.analisis-datos-cualitativos.pdf.

Herrera, Juan (2008). *Investigación cualitativa*. Recuperado el 29 de Setiembre, 2017, del sitio de Juan Herrera: https://juanherrera.files.wordpress.com/2008/05/investigacion-cualitativa.pdf.

*Is React a Framework?* (2015). Recuperado el 26 de Noviembre, 2017, de Medium: https://medium.com/@_cmdv_/react-is-not-a-framework-e25c28dcd78.

*Is React library or a framework?* (2016). Recuperado el 26 de Noviembre, 2017, del sitio Develoger: https://develoger.com/is-reactjs-library-or-a-framework-a14786f681a0.

*Javascript Frameworks you should know* (2017). Recuperado el 26 de Abril, 2018, de Apium hub: https://apiumhub.com/tech-blog-barcelona/top-javascript-frameworks/.

Lee, Raymond M. y Nigel G. Fielding (1998). *Computer Analysis and Qualitative Research*. SAGE Publications. ISBN: 0803974825.

Mikowski, Michael S. (2015). *RESTful APIs, the big lie*. Recuperado el 26 de Noviembre, 2017, del sitio Michael S. Mikowski: https://mmikowski.github.io/the_lie/.

Miles, M. y AM. Huberman (1994). *Qualitative Data Analysis*. Sage, Beverly Hills. ISBN: 9781452257877.

*Model One-to-Many Relationships with Document References*. Recuperado el 17 de Octubre, 2017, de MongoDB: https://docs.mongodb.com/manual/tutorial/model-referenced-one-to-many-relationships-between-documents/.

*Model One-to-Many Relationships with Embedded Documents*. Recuperado el 17 de Octubre, 2017, de MongoDB: https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-many-relationships-between-documents/.

Neuhaus, Jens (2017). *Angular vs. React vs. Vue: A 2017 comparison*. Recuperado el 26 de Noviembre, 2017, de Medium: https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176.

Puget, Jean Francois (2016). *The Most Popular Language For Machine Learning Is ...* Recuperado el 18 de Octubre, 2017, de IBM: https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science?lang=en.

Rodríguez, Gil y García (1996). *Metodología de la Investigación Cualitativa*. Ediciones Aljibe. Maracena. ISBN: 9788487767562.

Schettini, Patricia e Inés Cortazzo (2015). *Análisis de datos cualitativos en la investigación social*. Editorial de la Universidad Nacional de La Plata (EDULP).

Sommerville, Ian (2005). *Ingeniería del Software. Séptima edición*. PEARSON EDUCACIÓN S.A, Madrid. ISBN: 8478290745.

Stallman, R. (2004). *Software libre para una sociedad libre*. Madrid: Traficantes de Sueños, 2004. ISBN: 9781882114986.

Stallman, Richard. *Por qué el «código abierto» pierde de vista lo esencial del software libre*. Recuperado el 27 de Abril, 2018, del sitio de GNU: https://www.gnu.org/philosophy/open-source-misses-the-point.html.

Suárez Zendejas, Vicente. *Aproximación histórica de la investigación cualitativa*. Recuperado el 4 de Octubre, 2017, del sitio orion2020: http://orion2020.org/archivo/qualitas/03_historia_ql.pdf.

Voskoglou, Christina (2017). *What is the best programming language for Machine Learning?* Recuperado el 18 de Octubre, 2017, de Towards Data Science: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7.

*Why Quill*. Recuperado el 15 de Febrero, 2018 de Quill JS: https://quilljs.com/guides/why-quill/.

*¿Cuál es la diferencia entre el Software Libre y el Open Source?* (2016). Recuperado el 27 de Abril, 2018, de Genbeta: https://www.genbeta.com/a-fondo/cual-es-la-diferencia-entre-el-software-libre-y-el-open-source.

*¿Cuál es la diferencia entre el software "libre" y el software "gratuito"?* (2010). Recuperado el 27 de Abril, 2018, de Linux: `https://blog.desdelinux.net/cual-es-la-diferencia-entre-el-software-libre-y-el-software-gratuito/`.

*¿Qué es el software libre?* Recuperado el 26 de Abril, 2018, del sitio de GNU: `https://www.gnu.org/philosophy/free-sw.es.html`.

*¿Qué es una base de datos documental?* Recuperado el 8 de Octubre, 2017, de Amazon: `https://aws.amazon.com/es/nosql/document/`.

*¿Qué son las bases de datos documentales?* (2017). Recuperado el 8 de Octubre, 2017, de Smarter workspaces de Kyocera: `https://smarterworkspaces.kyocera.es/blog/las-bases-datos-documentales/`.

# Apéndice A

# Origen Investigación cualitativa

Sobre los orígenes de la investigación cualitativa no hay un consenso general de en qué momento se considera que se empezaron a usar métodos que estén enmarcados en esta forma de investigación. Sin embargo, se pueden observar incursiones hacia esta rama en varios momentos de la historia. "Según Gloria Pérez Serrano (. . . ) Los orígenes de la investigación cualitativa tienen antecedentes muy remotos en la cultura grecolatina y se conocen varios aspectos de esta metodología en las obras de Herodoto y Aristóteles." (citado por Alarcón Venegas et al. 2012)

En la era de la colonización, se puede identificar características cualitativas en la antropología cultural y evolucionista centrada inicialmente en el estudio de pequeños grupos humanos y tribus, la cual se interesa por la descripción y el análisis de culturas y comunidades con el fin de descubrir y explicar sus creencias y patrones de comportamiento.

> "Vidich y Lyman (1994), al analizar la historia de la investigación cualitativa desde la antropología y la sociología, consideran las siguientes etapas en su evolución: la Etnografía primitiva (1600), en la que tiene lugar el descubrimiento del otro; la Etnográfica colonial (1850), que continuará hasta la segunda guerra mundial, en la que los antropólogos escribían informes objetivos de sus experiencias de campo que reflejaban la influencia del paradigma positivista en su búsqueda de interpretaciones válidas y fiables en las que el otro era considerado un extraño, un extranjero. Es en esta época donde destaca la labor de los exploradores de los siglos XVII, XVIII y XIX. La Etnografía ciudadana (1900-1960), donde se estudia la etnografía del indio americano desde finales del XIX y principios del XX, la etnografía de los otros ciudadanos, estudios de comunidades y las etnografías sobre los inmigrantes americanos (desde comienzos del xix hasta los años sesenta). Otra etapa es la de la Etnicidad y asimilación, desde mediados del siglo xx hasta la década de los ochenta, y, por último, el momento actual, caracterizado por el cambio posmoderno." (Suárez Zendejas, citado por).

Según Alén Pérez (entrevista personal, 6 de Junio, 2017), las técnicas usadas por los antropólogos en los siglos XVII ,XVIII y XIX consistían en análisis de datos recopilados en viajes de investigación. Al llegar a una localidad desconocida se procuraba que los nativos los aceptaran, que les hablaran y les mostraran las cosas. ¿Qué método se utilizaba? No había una técnica. El investigador observaba todo, hablaba con todo el mundo, recolectaba datos, prestaba atención a los comportamientos. En sus comienzos los antropólogos no desarrollaron una metodología para estas investigaciones y se asentó la idea general de que el instrumento de investigación es el investigador.

Es recién a finales del siglo XIX y principios del XX los métodos cualitativos comienzan a emplearse de forma consciente.

> "Desde una perspectiva sociológica Bogdan y Biklen (1982), sitúan las raíces de la investigación cualitativa, dentro del seno de los Estados Unidos, en el interés sobre una serie de problemas de sanidad, asistencia social, salud y educación cuyas causas es preciso buscarlas en el impacto de la urbanización y la inmigración de grandes masas" (citado por Herrera, 2008, p. 8)

En las primeras décadas del siglo XX. En Estados Unidos se desarrolla en sociología la "Escuela de Chicago", principal desarrolladora de los métodos cualitativos de la época. Se empiezan a utilizar: la observación participante, la entrevista en profundidad y los documentos personales. Se realizan estudios de campo en antropología. En esa época de prosperidad en Estados Unidos y fuerte apoyo a las investigaciones el problema principal era la composición social de la población,la cual tenía familias provenientes de muchos países distintos. El mismo era un claro problema de cualidad, saber cuál era el motivo para que hubiera o no integración, investigar que generaba el rechazo interracial, intentar entender las barreras de las diferencias culturales. Entender la explicación de los procesos sociales era muy importante. Otra corriente que surge es la "Británica" desde 1920 en adelante la que se preocupa por el estudio de las normas de los grupos sociales.

Con la crisis del 29, el problema cambió y los principales intereses pasaron a ser las tasas de pobreza, hambruna, desempleo, migraciones,etc. La centralidad pasó a ser el enfoque cuantitativo, las estadísticas. A esto se le llama sociología cuantitativa y estuvo fuertemente aplicada al control poblacional por parte del estado.

> "Bogdan y Biklen (1982) señalan cuatro fases básicas en el desarrollo de la perspectiva cualitativa:
>
> 1. En primer lugar, una fase que se extiende desde finales del siglo XIX hasta los años 30, donde se presentan los primeros trabajos cualitativos y se consolidan técnicas como la observación participante, la entrevista en profundidad o los documentos personales.
> 2. Un segundo período va desde la década de los 30 hasta los años 50, donde se produjo un declive en la producción cualitativa.
> 3. Un tercer momento se produce en los años 60, época marcada por el cambio social y el auge de los métodos cualitativos.
> 4. Finalmente, un cuarto periodo iniciado en los años 60 donde se introducen nuevas perspectivas tanto sociológicas como antropológicas a partir de la evolución de su teoría social.
>
> En años recientes, asistimos a lo que Lincoln y Denzin (1994) llaman un quinto momento en la historia de la investigación cualitativa resaltando su carácter pluridisciplinar y multiparadigmático" (citado por Alarcón Venegas et al.,2012)

En la década de 1960 junto con los profundos cambios sociales de la época, la integración pasa a ser nuevamente un tema clave a estudiar y así se produce un auge de los métodos cualitativos. De ahí en adelante se experimenta una gran evolución debida a las nuevas perspectivas de la sociología y la antropología. Esta época, como afirman Rodríguez et al.(1996, p. 7), es de gran creatividad y se distingue por los

esfuerzos realizados para formalizar de manera sistemática y rigurosa los métodos y análisis de datos cualitativos con la aparición de toda una serie de textos al respecto (Bogdan y Taylor, 1975; Cicourel, 1964; Filstcad, 1970; Glasery Strauss, 1967; Lofland, 1971). El investigador cualitativo intenta realizar estudios cualitativos rigurosos de importantes procesos sociales. Conforme pasan los años se diversifican las estrategias de recogida y almacenamiento de la información y las herramientas informáticas comenzaron a ser usadas en la organización de datos y en análisis textual.

A partir del año 1990, el campo de los métodos cualitativos trascienden de las ciencias sociales para ser cada vez más utilizados en otras áreas investigativas. Desde sus inicios la investigación cualitativa ha sido atacada por los investigadores cuantitativos por su falta de validez. La ciencia se caracteriza por la validez externa, que implica que con la aplicación sistemática de un mismo método se debería obtener siempre el mismo resultado. Muchos autores han intentado dar rigor y exhaustividad a las técnicas cualitativas de análisis, tratando de que se aproximen en sistematización y fiabilidad a los procedimientos cuantitativos. Según Rodríguez et al.(1996, p. 2) el desarrollo de las metodologías cualitativas en la investigación se debe a diferentes áreas de las ciencias sociales, sobre todo la Sociología y la Antropología. Sin embargo hoy en día estos métodos traspasan estas disciplinas y se aplican cualquier investigación que conlleve el estudio de la relación de un individuo o grupo de individuos con su contexto.

**Uso de computadoras para la investigación cualitativa**

En 1966 aparece The General Inquirer, un programa de computadora para el análisis de contenido que puede considerarse, como el primero de los programas para el análisis de textos. Sin embargo el análisis que se realizaba con este recurso más bien respondía a un análisis cuantitativo del texto.

> "Este programa asignaba palabras de un documento a categorías específicas predeterminadas en un lexicón y calculaba las frecuencias de ocurrencia o co-ocurrencia de las palabras en cada categoría; también podía ayudar a la construcción del lexicón descomponiendo los textos en palabras, las cuales el investigador agrupaba sobre la base de su similitud semántica (Stone y otros, 1966)" (citado por Carvajal Llamas, 2001, p. 2).

Esta herramienta motivó la curiosidad de algunos investigadores cualitativos que se preguntaban si era posible obtener herramientas específicas que ayudaran en el proceso investigativo. En ese entonces la mayoría de los investigadores no sólo rechazaban el uso de este tipo de herramientas, sino que además consideraban que estas prácticas le quitaban validez a la investigación. El motivo de dicho rechazo se basa en la naturaleza .ªrtística"de esta rama investigativa. Al no haber un método específico para el análisis cualitativo, y ser más bien un paradigma bastante libre y variable según la investigación y el investigador particulares, no se consideraba posible realizar una sistematización de estos procedimientos.

> "Fue a mediados de la década de 1980 cuando los investigadores cualitativos "descubrieron" que los computadores podrían ser útiles en el análisis de su información, y unos pocos empezaron a usar procesadores de texto para ayudarse en esta tarea; sin embargo, las funciones de cortar y pegar seguían siendo las mismas, sólo que con algo de ahorro en el tiempo que demandaba la labor y en el espacio de almacenamiento

> del material (Wietzman y Miles, 1995:3-4)"(citado por Carvajal Llamas, 2001, p. 2).

Desde ese entonces el uso de las computadoras empezó a volverse un tanto más común entre los investigadores cualitativos, pero más que nada usando procesadores de texto para ayudarse en esta tarea. Recién en los años 90 se acelera el ritmo de expansión de los CAQDAS y se vuelven herramientas aceptadas por la mayoría de los investigadores, aunque aún al día de hoy son resistidos por una minoría.

# Apéndice B

# Proceso de desarrollo

## B.1. Metodología

Se realizó un desarrollo incremental por etapas, siguiendo una metodología similar a una ágil, comprendido en ciclos cortos de 2 o 3 semanas que llamaremos *sprints* durante este anexo. La elección de la metodología que se describe a continuación se realizó basándonos en la idea de dar prioridad a la calidad del resultado y a la validación temprana de cada funcionalidad.

## B.2. Herramientas utilizadas

Durante el proceso fue fundamental el uso de dos herramientas específicas Trello, Github y Google Drive. Se describe a continuación el uso dado por el equipo a dichas herramientas.

### B.2.1. Trello

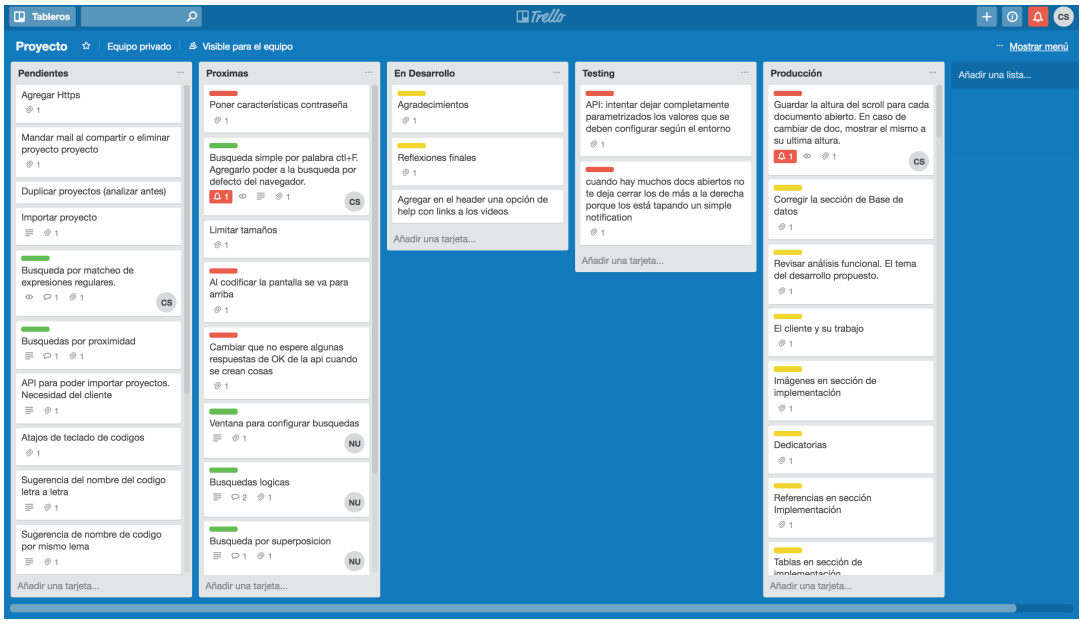



FIGURA B.1: Columnas de Trello.

Se utilizó la herramienta libre Trello[1] para el manejo de las actividades a realizar durante el proyecto, principalmente la administración de las funcionalidades a desarrollar por el equipo.

Se creó un tablero en Trello con cinco columnas con tarjetas como se ve en la figura B.1, donde cada una representaba una funcionalidad. Las columnas utilizadas fueron: Pendientes, Próximas, En Desarrollo, *Testing* y Producción. Se utilizaron también tarjetas de *bugs* y mejoras. Las tarjetas se etiquetaron según las entidades más implicadas en el desarrollo de la funcionalidad y se utilizó una extensión que permitía medir los tiempos para los cambios de estado de las tarjetas.

Al comenzar el proyecto, junto con la realización del análisis funcional, se generó un listado de funcionalidades que fue colocado en la primer columna de Pendientes. En la finalización de cada *sprint*, se priorizaron las funcionalidades a realizar y se colocaron en la segunda columna. Cada miembro del equipo tomaba algunas de las tarjetas para desarrollar y se las asignaba. Cuando comenzaba el desarrollo de la tarjeta, la misma se movía a la columna En Desarrollo y al finalizar y subir se pasaba a *Testing*. Luego de la verificacion y validacion se pasaba la tarjeta a la columna Producción.

Durante el transcurso del proyecto surgieron algunas nuevas funcionalidades que fueron evaluadas y en algunos casos, se agregaron a la columna de Pendientes. Como dijimos anteriormente el tablero no solo contenía funcionalidades, sino que también los distintos *bugs*, que mantenían el mismo flujo que las funcionalidades. Sobre el final del proyecto también se utilizó el tablero para coordinarnos en la escritura de la documentación del informe.

### B.2.2. Github

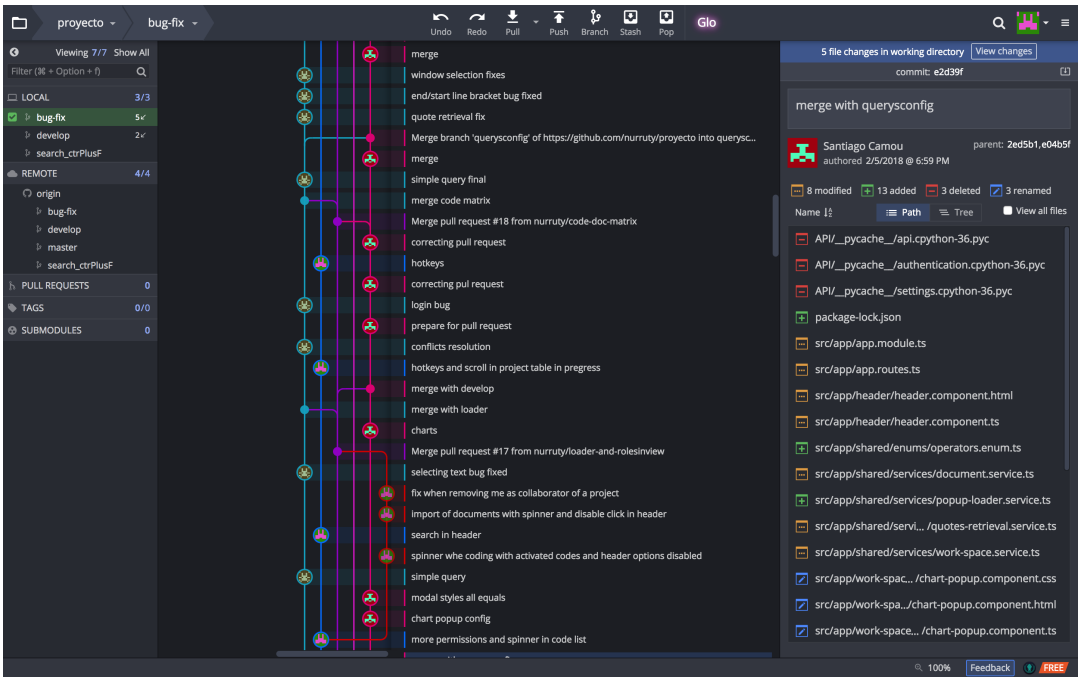



FIGURA B.2: Ramas.

[1] trello.com

Se usó Github[2] como repositorio y mantenimiento de versiones del código fuente. Se utilizó la rama *develop* como rama principal, en la cual se fueron agregando las funcionalidades desarrolladas y probadas. Se crearon durante el proceso subramas específicas para el desarrollo de conjuntos de funcionalidades relacionadas, que eran utilizadas en general por un miembro del equipo para el desarrollo de sus tarjetas asignadas. Esta estructura permitió el orden, la posible colaboración entre los desarrolladores en distintas funcionalidades y el mantenimiento de una versión estable de la herramienta en la rama principal.

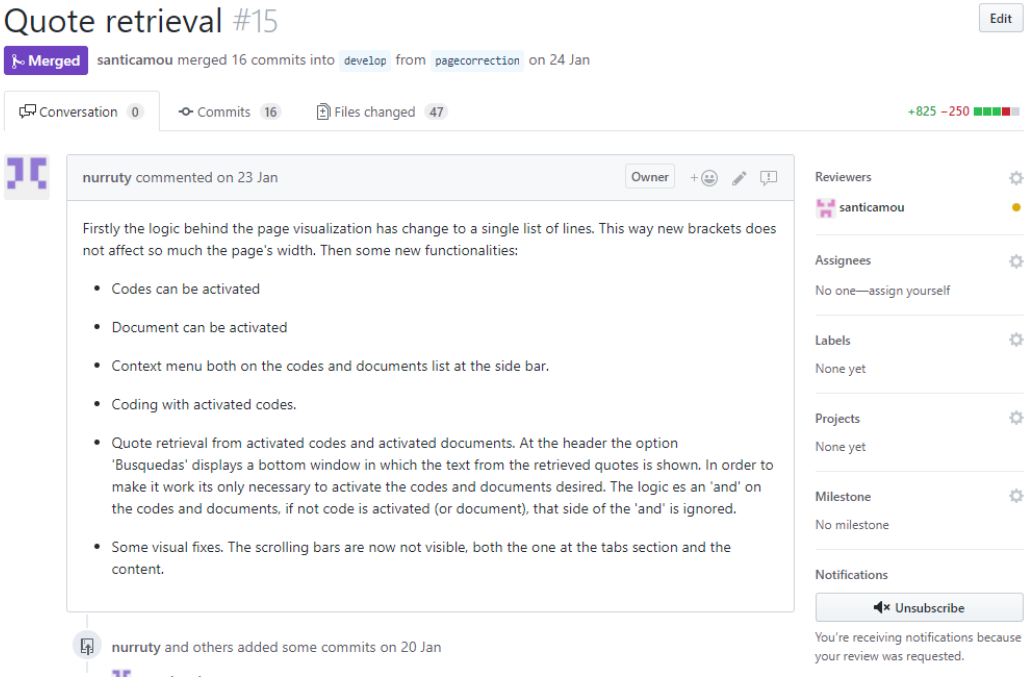



FIGURA B.3: *Pull Request*.

Para la implementación de cada subconjunto de funcionalidades se creó como ya se comentó, una nueva rama dedicada. Los *bugs* fueron agregados a ramas de funcionalidades o ramas específicas para realizar correcciones. Al finalizar la implementación de dichas funcionalidades en la rama se hacía un *Pull Request* de la misma solicitando el *merge* con *develop* e indicando en un comentario las nuevas funcionalidades, mejoras y reparaciones que se desarrollaron, como se ve en la figura B.3. Otro miembro del equipo, de preferencia que no hubiera realizado ningún aporte a dicha rama se encargaba de la verificación y validación. De encontrarse errores o posibles mejoras, agregaba un comentario en el *Pull Request* e informaba, para que el desarrollador de dichas funcionalidades realizara los cambios necesarios, actualizara el *Pull Request* y el proceso de verificación se repetía. Cuando todo esto se validaba, se aceptaba el pedido quedando las funcionalidades en la rama *develop*.

En momentos elegidos por el equipo, las funcionalidades en la rama principal se subían a Heroku (ver Apéndice F) para ser mostradas a la cliente, y a otros interesados.

[2]github.com/

### B.2.3. Google Drive

Como repositorio para documentación se utilizó Google Drive[3], organizando los archivos en carpetas compartidas entre los miembros del equipo y la tutora, lo cual facilitó el ida y vuelta para las correcciones.

[3]drive.google.com

# Apéndice C

# Manual de Usuario

Te damos la bienvenida al manual de usuario de Qode, una herramienta web de uso libre para el análisis cualitativo de datos.

## C.1. Inicio de sesión

Para acceder a la aplicación, es necesario estar registrado. El inicio de sesión se realiza completando los datos usuario y contraseña en el formulario que aparece en pantalla como en la figura C.1. También se permite el ingreso a la aplicación con las credenciales de su cuenta de google. Posteriormente, el sistema redirige a la página de "Mis proyectos".

En caso de olvidar la contraseña, se debe hacer click en el botón 'Olvidó su contraseña?', el cual redirige a la siguiente pantalla. Se enviará un email al usuario solicitando el ingreso de una nueva contraseña.

## C.2. Mis Proyectos

**Creación de proyectos**

Para crear un proyecto se debe ingresar un nombre y opcionalmente una descripción, en los campos correspondientes como se muestra a la figura C.2.

**Tabla de Mis Proyectos**

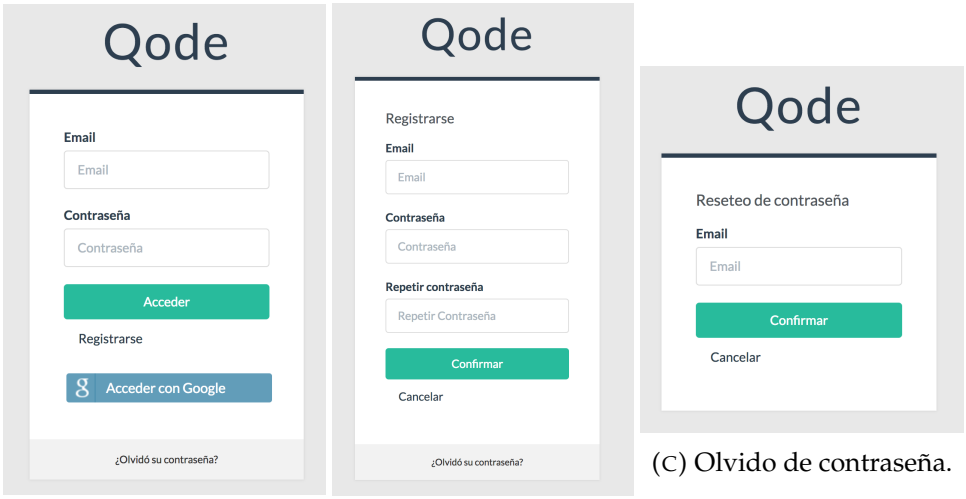
En esta tabla se listan los proyectos creados por el usuario logueado y los proyectos en los que se encuentra como colaborador. La tabla permite filtrar por nombre de proyecto y ordenar de forma ascendente o descendente por sus columnas: Nombre, Propietario y Fecha de su última actualización realizada por su creador o cualquiera de sus colaboradores. El usuario logueado solo puede borrar los proyectos que creó.

Al hacer click en una fila de la tabla se despliega la información del proyecto en el área a la derecha de la pantalla. Haciendo doble click en la fila se accede al proyecto correspondiente.

**Información de Proyecto**

Como se muestra en la figura C.3, la barra lateral derecha de la pantalla contiene la siguiente información del proyecto:

- **Edición de descripción** Es posible modificar la descripción ingresada en el momento de la creación.
- **Información** Se observan los datos básicos del proyecto. El usuario creador, su fecha de creación, la fecha de su última actualización y que usuario realizó la misma. Se muestra una lista de los colaboradores del proyecto y su rol.




(A) Inicio de sesión. (B) Registro de usuario. (C) Olvido de contraseña.

FIGURA C.1: Acceso al sistema.


FIGURA C.2: Workspace.

- **Compartir proyecto** Los proyectos pueden ser compartidos a otros usuarios de la aplicación. Se debe ingresar el email y el rol que se le otorgará a dicho usuario en ese proyecto. Los roles son de Lectura y/o escritura, y pueden ser modificados o revocados posteriormente.
- **Acceder** Acceso al proyecto seleccionado que redirige a el ambiente de trabajo o workspace C.4.

## C.3. Workspace

**Header**
El header de la página provee varios menús separados por áreas.

- **Proyecto**

  Permite compartir el proyecto al igual que se hace desde la pantalla "Mis proyectos".

- **Documentos**

  Permite importar documentos desde el sistema de archivos de la PC. Son soportados los formatos .pdf, .docx, .txt y .rtf.

- **Códigos**

  Se dispone de dos funcionalidades: crear un código nuevo (se despliega una ventana para edición), e importar códigos de otro proyecto, la cual permite al usuario importar códigos de otro proyecto en el cual participa como colaborador o creador, sin importar qué permisos tiene sobre el mismo.


FIGURA C.3: Información de proyecto seleccionado.

- **Búsquedas**

  Hay una sola opción de búsqueda: las simples, que recuperan citas asociadas a algún código. Se puede elegir tanto para los códigos como para los documentos entre recuperar las citas asociadas a cualquiera de ellos o solo a un conjunto seleccionado mediante la activación de los mismos, como se ve en la figura C.5. Los resultados se despliegan como se ve en la figura C.5 en un panel que aparece en la parte baja de la pantalla, desde donde se puede ver el listado de citas, y se puede hacer click en el ícono que aparece en la esquina superior izquierda de la referencia a la cita para que la misma se muestre en el área de lectura del documento abierto.

**Análisis**

Hay dos funcionalidades de análisis basado en la codificación de citas realizada. Las dos consisten en la obtención de ciertos resultados numéricos que expresan la relación entre códigos y documentos o de códigos entre sí. Los resultados que se pueden elegir son: la cantidad de ocurrencias de citas asociadas a cada código para cada documento, y la cantidad de coocurrencias de citas a asociadas a distintos códigos en un mismo documento. En ambos casos los resultados se pueden presentar en forma de tabla o de gráfica. Las figuras C.7, C.8, C.9 muestran distintos tipos de representaciones. Las gráficas se pueden guardar como imagen mediante botón derecho sobre las mismas.

Además hay tres botones que nos llevan a las pantallas anteriores. Los botones "Qode", en la esquina izquierda, y el botón "Mis proyectos" llevan a la pantalla de administración de proyectos del usuario. Mientras que el botón "Logout" cierra la sesión del usuario y despliega la pantalla inicial de inicio de sesión.

**Lista de documentos**

Se encuentra el listado de los documentos que han sido importados al proyecto, como se observa en la figura C.10. Cada documento indica la cantidad de citas que

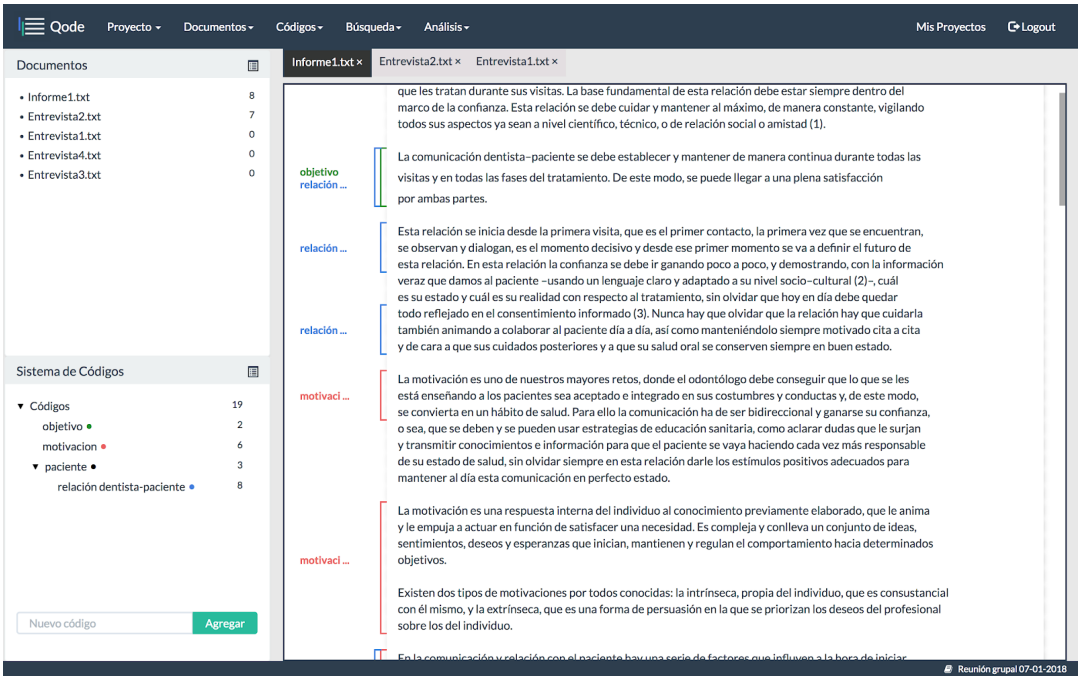



FIGURA C.4: Workspace.

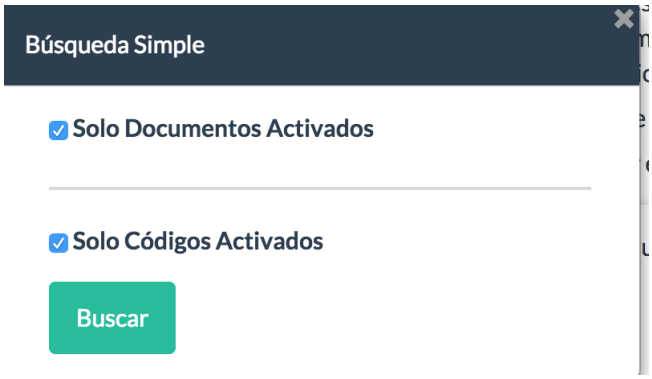



FIGURA C.5: Búsqueda simple.

tiene asociadas. Mediante click derecho sobre el nombre de los documentos se permiten abrir,borrar,activar o desactivar el documento y editar sus atributos. También mediante click sobre el ícono en la esquina superior derecha del panel de lista de documentos se puede activar o desactivar todos los documentos a la vez. Otra forma de activar o desactivar un documento individual es apretando CTRL + click sobre el nombre del mismo.

**Sistema de códigos**

Como se muestra en la figura C.11 Se encuentra el listado de los códigos pertenecientes al proyecto. El mismo permite organizarlos en forma jerárquica mediante "drag and drop". La estructura jerárquica de los códigos se aplica solamente para la visualización en el listado. Para las acciones de codificación y análisis todos los códigos son iguales e independientes entre sí. Cada código indica la cantidad de citas asociadas que tiene. Al igual que para los documentos, hay un ícono para activar o desactivar todos los códigos a la vez, y en caso de querer realizar estas acciones sobre un código individual se puede hacer CTRL + click sobre el nombre del mismo. También se puede hacer click derecho sobre el nombre de un código para las

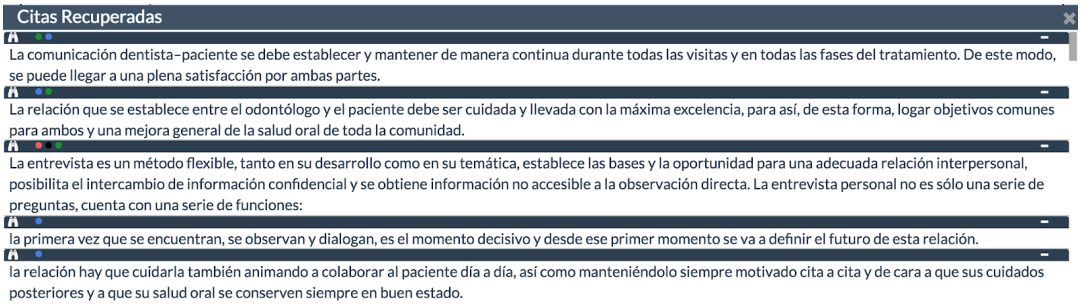



FIGURA C.6: Citas recuperadas

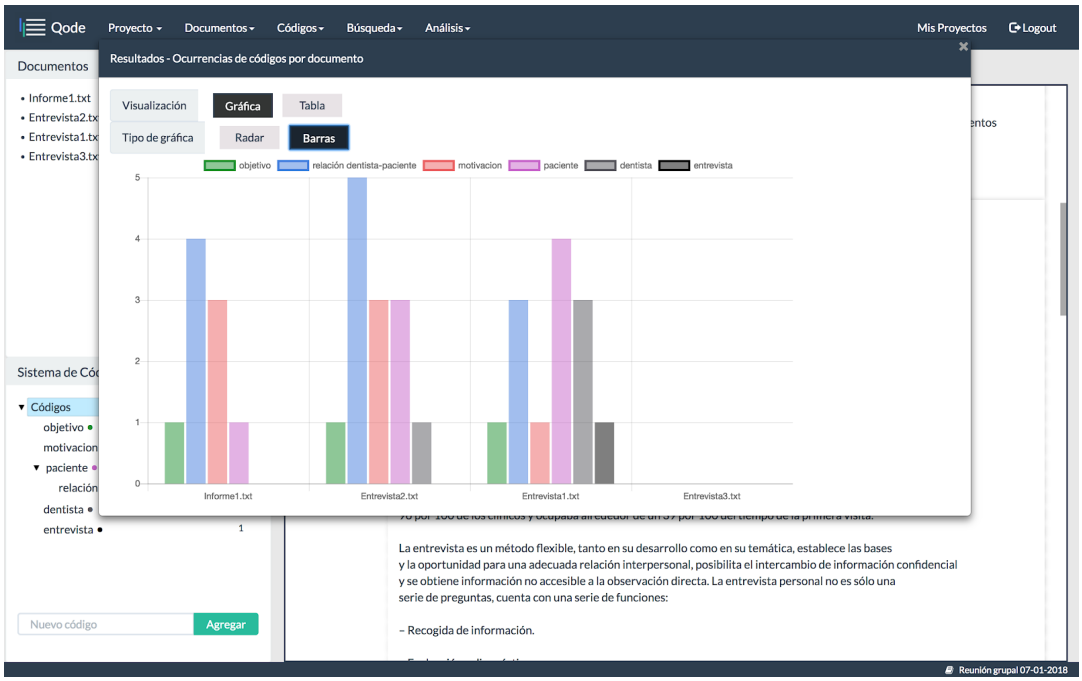



FIGURA C.7: Ocurrencias de códigos por documento

funciones de activación,desactivación, borrado o edición del código.

**Documentos abiertos**

Los distintos documentos abiertos se muestran en la zona central de la pantalla, mediante una lista de tabs que permiten seleccionar el que se desea visualizar y un área de lectura donde se ve el texto correspondiente al documento seleccionado.

En esta área de lectura si se selecciona una porción de texto haciendo click derecho se despliega un menú de opciones, entre las cuales están: codificar: despliega una ventana donde se pueden agregar códigos y un memo a la cita. codificar con códigos activados: se hace uso de los códigos activados para crear una cita automáticamente con ellos. copiar el texto seleccionado.

Las citas creadas generan un sistema de corchetes que ayudan en la visualización. Cada cita es representada por un corchete por cada código asociado, o un solo corchete en caso de no tener códigos asociados. Al hacer click en uno de los corchetes correspondientes a la cita se abre una ventana para editar los códigos y el memo asociados a la cita o borrarla. Los corchetes que se corresponden con un código tienen su color correspondiente y muestran el nombre del mismo a la izquierda.

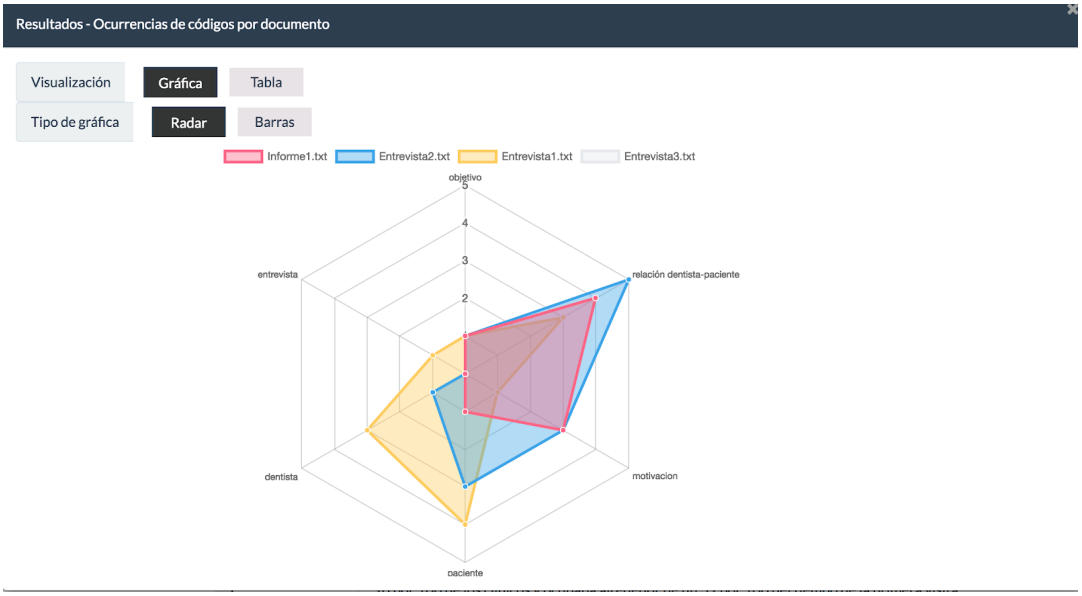



FIGURA C.8: Ocurrencias de códigos por documento 2


| Documentos\Códigos | objetivo | relación dentista-paciente | motivacion | paciente | dentista | entrevista |
|---|---|---|---|---|---|---|
| Informe1.txt | 1 | 4 | 3 | 1 | 0 | 0 |
| Entrevista2.txt | 1 | 5 | 3 | 3 | 1 | 0 |
| Entrevista1.txt | 1 | 3 | 1 | 4 | 3 | 1 |
| Entrevista3.txt | 0 | 0 | 0 | 0 | 0 | 0 |

FIGURA C.9: Ocurrencias de códigos por documento 3


FIGURA C.10: Lista de documentos.


FIGURA C.11: Lista de códigos.

# Apéndice D

# Pseudo código cálculo posición corchetes

La idea del algoritmo es encontrar, para ua nueva cita, qué columnas (de la tabla que define la estructura del texto) ocupará el o los corchetes que describen a la cita. Recordando, un corchete es un conjunto continuo vertical de celdas de la estructura tabular que hacen referencia a un código dentro de una cita. El cuadro D.1 representa el ejemplo de un conjunto de citas visto en el capítulo 4.

| | | | |
|---|---|---|---|
| codigo C - cita 3 - l1 | codigo B - cita 2 - l1 | codigo A - cita 1 -l1 | texto l1 |
| codigo C - cita 3 - l2 | codigo B - cita 2 - l2 | | texto l2 |
| codigo C - cita 3 - l3 | codigo B - cita 2 - l3 | | texto l3 |
| codigo C - cita 3 - l4 | codigo B - cita 2 - linea 4 | codigo D - cita 4 -l4 | texto l4 |

CUADRO D.1: Estructura tabular de texto y cita

El siguiente pseudocódigo describe cálculo para una nueva cita. La función devuelve la primera columna libre desde la cual pueden colocarse el o los corchetes que le corresponden. Se evalúan tantas columnas como códigos tenga la cita hasta encontrar aquellas columnas contiguas que, según la línea inicial y final de la cita, no haya otra (cita) que tenga un corchete que colisione con el que se quiere colocar.

**Buscar primer columna libre cita**(CITA, DOC):

```
let columna = 0
let exito = false
let largo = CITA.cantCorchetes
while  (not exito) and (columna less than or equals to MAXCOLUMNAS) do
    while not exito do
        let i = 0
        let iterExito = true
        while (iterExito) and (i less than largo) do
            iterExito = EstaLibreColumna(column + i)
            i++
        end while
        exito = iterExito
        if (not exito) then
            columna += i
        end if
    end while
```

```
  end while
  return columna

EstaLibreColumna(CITA, DOC, columna):

  let tomada = false
  let iter = 0
  while not tomada and iter less than CITA.cantLineas do
      ini = CITA.lineaIni
      fin = CITA.lineaFin
      citasAnteriores = DOC.citas[column]          ▷ Citas ya ubicadas en la columna
      for ( i=0 : Largo(citasAnteriores) - 1 ) do   ▷ Largo devuelve el largo de la lista
          iniAnt = citasAnteriores[i].lineaIni
          finAnt = citasAnteriores[i].lineaFin
          if Intersecta(ini, iniAnt, fin, finAnt) then
              tomada = true
          end if
      end for
  end while
  return tomada


Intersecta(ini, iniAnt ,fin, finAnt):

  return iniAnt less than or equals to fin and ini less than or equals to iniAnt
```

Con el pseudocódigo anterior se logra ubicar a la nueva cita en la estructura en pantalla. Dicho cálculo se realiza tanto al crear una cita nueva, como cuando se traen todas las existentes en la base de datos.

# Apéndice E

# Manual de Instalación

Los siguientes pasos describen como descargar Qode desde su respositorio oficial y como dejarlo funcionando en un entorno local.

1. Entrar al sitio oficial de nodeJS (nodejs.org) y descargar la una versíon mayor a la 8.3. En sistemas operativos tipo unix o mac se puede instalar mediante línea de comando usando el paquete de instalación correspondiente al sistema.

2. Descargar e instalar Python en una versión mayor a 3.5 desde el sitio oficial de Python (python.org/downloads/).

3. Descargar MongoDB desde su sitio oficial (mongodb.com/download-center).

4. Abrir un terminal y descargar el código fuente desde GitHub (github.com) mediante el siguiente comando:

   ```
   $ git clone https://github.com/nurruty/qode.git
   ```

5. Ingresar a la carpeta *API* e instalar los paquetes de Python necesarios:**

   ```
   $ cd qode
   $ cd API
   $ pip install requirements
   ```

6. Ingresar a la carpeta *src/app* e instalar las dependencias de Angular descritas en el archivo *packages.json*:

   ```
   $ cd ..
   $ cd src/app
   $ npm install
   ```

7. Modificar el archivo *config.py* ubicado en la carpeta API. En este se configuran varios campos correspondientes a la validación de los *tokens* que se reciben para autenticar al usuario, también se agrega la dirección en donde se ejecuta la base de datos en caso de usar una base externa, sino se pone nada la base que se ejecuta es la correspondiente a los datos almacenados en la carpeta /API/data/db.

8. Desde la carpeta principal de Qode acceder a src/environments, en este directorio se encuentran dos archivos *environment.prod.ts* y *environment.ts*.Los mismos contienen las configuraciones correspondientes a los entornos de desarrollo y producción respectivamente. Estas configuraciones implican las direcciones en las que se despliegan tanto el componente de *front-end* como la API, y luego varios datos correspondientes a la comunicación con Auth0. Al ejecutar el programa se usa uno u otro en función de cómo se realice la ejecución.

9. Configurados los archivos anteriores, se debe ejecutar el comando:

```
$ ng serve
```

10. Ingresar a la dirección http://localhost:4200

Los pasos anteriores describen como iniciar Qode en un entorno local. Si se quisiera desplegar el sistema en producción se debería configurar en primer lugar se debería configurar un servidor específico para el funcionamiento de la API, con las configuraciones de Auth0 correspondientes.

Por otro lado el código del *front-end* debe ser compilado ejecutando:

```
$ ng build --env=prod
```

Con este comando se crea una carpeta dist. La misma contiene todos los archivos necesarios para que el sistema funciones en producción. Las configuraciones específicas de este entorno deben realizarse en el archivos *environment.prod.ts*. A su vez, debe levantarse un servidor web específico que responda con la página *index.html* ubicada dentro de la carpeta *dist*.

**

Existe una excepción en el paso 5. En Windows puede suceder que al instalar las dependencias de Python mediante el comando **pip install requirements** se de un error queriendo instalar la librería *pyhton-jose*.

En ese caso se debe seguir el siguiente procedimiento:

- Si no tiene compilador c++ instalado descargar e instalar de:
  http://landinghub.visualstudio.com/visual-cpp-build-tools
- Abrir ventana de comandos con permisos de administrador y ejecutar el archivo *vcvarsall.bat* que aparece en la carpeta donde se instaló el paquete anterior.
- Ejecutar el comando:

```
$ set CL=-FI"%VCINSTALLDIR%\INCLUDE\stdint.h"
```

- Luego ejecutar:

```
$ pip install pycrypto
```

- Finalmente:

```
$ pip install python-jose
```

# Apéndice F

# Instructivo para realizar el *deploy* en Heroku

## F.1. Componente de *front-end*

Para realizar el *deploy* del servicio que devuelve la página Angular de Qode en Heroku se debe contar previamente con las siguientes herramientas instaladas:

- heroku CLI [1]
- git [2]

Con lo anterior instalado se debe seguir el instructuivo [3] de la página oficial de Heroku para enlazar una carpeta local con el código fuente, con el respositorio git remoto de Heroku.

Luego de enlazada la carpeta local, se deben seguir los siguientes pasos:

1. El archivo *environement.prod.ts* tiene que lucir de la siguiente forma:

   export const environment =

   ```
   {
     production: true,
     apiUrl: 'https://<APINAME>.herokuapp.com/',
     frontUrl: 'https://<APPNAME>.herokuapp.com/',
     clientID: '<AUTH0_CLIENT_ID>',
     domain: '<AUTH0_DOMAIN>.herokuapp.com/auth0',
     domainSocial: '<APPNAME>.auth0.com'
   }
   ```

2. Abrir un terminal en la carpeta con el código fuente. Compilar la aplicación mediante el comando:

   ```
   $ ng build --env=prod
   ```

3. Copiar la carpeta *dist* generada, a la carpeta desde donde se va a realizar el *deploy*.

[1] devcenter.heroku.com/articles/heroku-cli
[2] git-scm.com
[3] devcenter.heroku.com/articles/git

4. en la carpeta desde donde se va a realizar el deploy crear un archivo llamado *static.json* con el siguiente contenido:

```
{
  "root": "dist/",
  "https_only": true,
  "routes": {
    "/**": "index.html"
  },
  "proxies": {
       "/auth0/": {
      "origin": "https://webappqode.auth0.com/"
    }
  },
  "headers": {
    "/": {
      "Cache-Control": "no-store, no-cache"
    }
  }
}
```

5. Desde línea de comandos:

```
$ heroku plugins:install heroku-cli-static
$ heroku buildpacks:set https://github.com/home
/heroku-buildpack-static
$ git commit -a -m 'mensaje'
$ heroku static:deploy
$ heroku ps:scale web=1
```

## F.2. Componente de *back-end*

Para liberar el *back-end* se debe realizar en una carpeta separada del *front-end* ya que son aplicaciones distintas. La conexión con el directorio git remoto se hace también por separado siguiendo los pasos del instructivo de la página oficial utilizados en el punto anterior.

1. La carpeta debe contener, ademas de todos los archivos encontrados en la carpeta API del código fuente, un archivo llamado *config.py* con los siguientes datos:

```
MONGO_URI = 'mongodb://proyecto2018:proyecto2018@ds131329.mlab.com:3132
heroku_h3gtgnjg'
AUTH0_DOMAIN = '<AUTH0_DOMAIN>.auth0.com'
API_AUDIENCE = 'https://<APINAME>.herokuapp.com/'
ALGORITHMS = ["RS256"]
RULE_CLAIM = 'https://myapp.example.com/email'
```

2. En la carpeta debe haber un archivo llamado Procfile (sin extensión) con el siguiente contenido:

```
web: gunicorn api:APP
```

3. En línea de comandos parado en la carpeta, ejecutar:

```
$ git commit -a -m 'mensaje'
$ git push heroku master
```

# Apéndice G

# Software Libre

El *software* libre, representa todos aquellos que respetan las libertades esenciales del usuario definidas por la FSF (Free Software Foundation) [1], fundación creada en 1985 por Richard Stallman para defender las libertades en el mundo tecnológico.

Este movimiento ha hecho campaña por la libertad de los usuarios de ordenador desde 1983. En 1984 se inició el desarrollo del sistema operativo libre GNU, para poder evitar el uso de sistemas operativos que no son libres y que niegan la libertad a los usuarios. Durante los años ochenta se desarrolla la mayor parte de los componentes esenciales del sistema GNU [2], y se diseña la Licencia Pública General de GNU (GNU GPL, por sus siglas en inglés) para usarla en la distribución de dichos componentes; una licencia diseñada específicamente para proteger la libertad de todos los usuarios de un programa. Sin embargo, no todos los usuarios y programadores de *software* libre estaban de acuerdo con los objetivos del movimiento. En 1998 una parte de la comunidad se bifurcó y dió inicio a una campaña para promover el *open source* (código abierto). La expresión se propuso originalmente para evitar un posible malentendido con el término *free software* (*software* libre), pero pronto se asoció con posiciones filosóficas diferentes a las del movimiento.
Las cuatro libertades esenciales de los usuarios definidas por la FSF:

1. La libertad de ejecutar el programa como se desea, con cualquier propósito.
2. La libertad de estudiar cómo funciona el programa, y cambiarlo, para lo cual es necesario el acceso al código fuente.
3. La libertad de redistribuir copias para ayudar a otros.
4. La libertad de distribuir copias de sus versiones modificadas a terceros, que permite a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es tambien condicion necesaria.

Según la fundación, las libertades, tienen que ser aplicadas en todo el código de un programa. De esta manera, si se crea *software* a partir de otro, hay que asegurarse que tanto la base sobre la que se trabaja como las modificaciones realizadas respeten estas libertades. El *software* producido a partir de *software* libre debería ser libre también. Sin embargo, aunque siga las libertades antes mencionadas, no tiene por que ser gratuito, ya que puede conservar su carácter libre pero ser distribuido de manera comercial.

[1] fsf.org
[2] gnu.org

### Diferencias con el código abierto

Cuando nos encontramos ante un *software* no propietario, podemos encontrar entonces dos modalidades: *software* libre y de código abierto. Estas licencias están estrechamente relacionadas y se suelen utilizar indistintamente.

Como se menciona en (*¿Cuál es la diferencia entre el Software Libre y el Open Source?* 2016) y (*¿Cuál es la diferencia entre el software "libre" y el software "gratuito"?* 2010), ambas expresiones describen casi la misma categoría de *software*, pero representan puntos de vista basados en valores fundamentalmente diferentes. El código abierto es una metodología de programación, el *software* libre es un movimiento social. El segundo se centra más en las cuestiones éticas relacionadas con el mundo del *software* y respeto esencial por la libertad de los usuario, mientras que el primero prioriza el apartado técnico de la aplicación. El código abierto, es una licencia más enfocada hacia los beneficios que los usuarios pueden obtener por tener acceso al código fuente de las aplicaciones para su posterior mejora y actualización (*Diferencias entre software libre, de código abierto y gratuito* 2018).

En la práctica, como compara Richard Stallman en (*Por qué el «código abierto» pierde de vista lo esencial del software libre*) y (Stallman, 2004) el código abierto es menos estricto, por lo que todo *software* libre se puede calificar como código abierto, aunque no todo el *software* de código abierto tiene por qué ser libre. Algunas licencias de código abierto son demasiado restrictivas, por lo que no se las puede considerar como libres, por ejemplo si su licencia no permite hacer versiones modificadas y utilizarlas de forma privada.

Básicamente el *software* sibre tiene diferencias de índole filosóficas con el de código abierto, sin embargo, ambos han permitido el desarrollo de *software* de gran calidad y con muchísimas mas libertades de las que el software privativo podría otorgar (*Diferencias entre Software Libre y Open Source* 2014).

### Licencias

Si la idea es que cualquiera pueda ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el *software*, entonces, ¿por qué resulta necesaria la utilización de licencias?

Estas licencias surgen para regular y proteger proyectos de software libre. Sin esta regulación, por ejemplo, una empresa puede tomar un proyecto de *software* libre y cubrirlo bajo una licencia restrictiva o privativa. La diferencia entre el libre y el privativo radica en el tipo de licencias que estos utilizan.

En el caso de Qode se eligió una licencia poco restrictiva como Apache License v2 [3] de la Apache Software Foundation.

La licencia Apache es permisiva ya que no exige que las versiones modificadas del *software* se distribuyan usando la misma licencia, ni que se tengan que distribuir como *software* libre/código abierto. Sólo exige que se mantenga un aviso que informe a los receptores que en la distribución se ha usado código con la licencia Apache.

Se añadieron dos archivos en el directorio principal del *software* del proyecto en el repositorio de Github.

[3] .apache.org/licenses/LICENSE-2.0