MIA
Desarrollo de una aplicación para pacientes con rinitis alérgica

Proyecto de grado de Ingeniería en Computación

Facultad de Ingeniería

Noviembre 2021

Autoras:

Tamara Súarez
Natalia Campiglia

Tutoras:

Libertad Tansini
Regina Motz

Agradecimientos

Queremos dedicarle un especial agradecimiento a Ángeles Beri, Alejandra Leal, Carina Almirón, Libertad Tansini, Mariana Bonifacino, Patricia Indarte, Rosario Eugui, Regina Motz y Ximena Martínez, quienes desde su área de conocimiento aportaron todo lo que estaba a su alcance, haciendo posible este proyecto.

MI ALERGIA

Aplicación para rinitis alérgica.

Tamara Suarez, Natalia Campiglia

Resumen

El objetivo del presente proyecto es diseñar e implementar un Producto Mínimo Viable (MVP) de una plataforma que permita ayudar a los pacientes con rinitis alérgica en el seguimiento de su condición, para identificar los desencadenantes de sus alergias y colaborar con el tratamiento adecuado. Es por esto que el proyecto tiene dos grandes pilares, por un lado los reportes de polen y esporas de hongos, que le permitirán al paciente tomar los recaudos necesarios ante altas concentraciones, y por otro lado un diario de síntomas, que le permitirá tener un historial de cómo se ha sentido, pudiendo en conjunto con un médico tratante llegar a un tratamiento más adecuado, dado que muchas veces sucede que al momento de un paciente acudir a su control médico, no recuerda en detalle cuántos días se sintió mal, o qué tan mal se sintió, en estos casos acudir con un registro a una consulta le da más contexto al especialista. También cuando se ingrese un nuevo evento en el diarios de síntomas, en caso de que sea pertinente, se sugerirán acciones a tomar por parte del paciente.

Este documento describe los detalles del proyecto de grado. El mismo se enmarcó en el proyecto semillero [1], financiado por el Espacio Interdisciplinario (EI) de la Universidad de la República (`https://www.ei.udelar.edu.uy/`), área que busca apoyar y financiar proyectos que requieran de diferentes disciplinas para su elaboración. El presente proyecto de grado se desarrolla bajo el semillero *Hiperreactividad respiratoria en Montevideo: Integración de factores ambientales para una medicina de precisión* [2], conformado por las disciplinas de la biología, medicina y computación. La descripción completa del equipo del Proyecto Semillero se presenta en el Apéndice A.

Con el acompañamiento del grupo de médicas y el grupo de palinólogas, se logró desarrollar una aplicación para teléfonos móviles, llamada MIA, por el acrónimo de Mi Alergia, en la cual se difunden los datos aeropalinológicos diarios y se compila la sintomatología de los pacientes. Esta aplicación está disponible para ser descargada en las tiendas digitales de Android y Apple. El lanzamiento inicial de la plataforma

se probó con un reducido grupo de pacientes del Hospital Pasteur, en un piloto realizado a lo largo del mes de noviembre del 2020 y hoy en día cuenta con un amplio público. También se realizó una página web para la presentación del proyecto y un administrador web para que terceros dentro del equipo puedan ingresar datos de las mediciones, que luego son procesados por el sistema. El código de MIA se encuentra de forma pública en `https://github.com/mialergia`.

Índice general

Índice de figuras

Índice de cuadros

Siglas

API Application Programming Interface. 3, 4, 36, 50, 62, 71, 77

MVP Producto Mínimo Viable. II, 5, 64, 95

ORM Object Relational Mapping. 36

REST Representational state transfer. V, 3, 62

Glosario

Android Sistema operativo móvil basado en núcleo Linux y otros softwares de código abierto. II, 2, 3, 6, 8, 11, 33, 36, 59–61, 65, 83, 84

Angular Es un framework de código abierto desarrollado por Google para facilitar la creación y programación de aplicaciones web. V, 4, 37, 69

API REST Una API REST es un estilo arquitectónico para una interfaz de programa de aplicación (API) que usa solicitudes HTTP para acceder y usar datos. Esos datos se pueden utilizar para OBTENER (GET), PONER (PUT), POSTAR (POST) y ELIMINAR (DELETE) tipos de datos, lo que se refiere a la lectura, actualización, creación y eliminación de operaciones relacionadas con los recursos. 3, 36, 62, 67

App Store Tienda de aplicaciones móviles para el sistema operativo iOS. 10, 33

backend La parte del sistema que se encarga de almacenar y comunicar los datos entre el frontend y la base de datos. 3, 4, 36, 37, 44, 45, 50, 53, 59, 62, 65, 67, 70–72, 79, 94

deploy El proceso por el que se hace público o se pone a disposición de los usuarios un producto que estaba en fase de desarrollo o pruebas. 64, 70

Django Es un framework de Python libre y de código abierto que sigue el patrón de arquitectura model–template–views. VIII, 3, 36, 51, 59, 61, 62, 67, 79

endpoint Un API endpoint (punto final) es el punto de entrada en un canal de comunicación cuando dos sistemas están interactuando. El punto final puede verse como el medio desde el cual la API puede acceder a los recursos que necesitan desde un servidor para realizar su tarea. 62, 72, 73, 75

frontend La parte del sistema que el usuario ve e interactúa. 3, 4, 44, 45, 53, 60, 61, 63, 71, 74, 76, 79, 94

iOS Sistema operativo móvil creado y desarrollado por Apple, Inc. VIII, 2, 3, 6, 36, 59–61, 83, 84, 95

palinología La palinología es una disciplina de la Botánica, dedicada al estudio del polen y las esporas. 36, 39, 40, 46, 51, 54, 56, 78, 81, 83

Play Store Tienda de aplicaciones móviles para el sistema operativo Android. 33, 95

Prick prick es un test, también conocido como pruebas cutáneas de alergia, que consiste en una serie de pruebas que se realizan en la piel con el fin de identificar sustancias que puedan causar una reacción alérgica en el paciente. Estas pruebas están consideradas como el método ideal para diagnosticar posibles alergias.. 45, 49, 51, 85, 92

React Es una librería de JavaScript de código abierto para desarrollar interfaces de usuario. Creada y desarrollada por Facebook, Inc. V, 60

React Native Es un framework de React creado por Facebook, Inc. Utilizado para desarrollar aplicaciones para Android, Android TV, iOS, macOS, tvOS, Web, Windows y UWP. V, VIII, 3, 36, 59–61, 71, 79

Capítulo 1

Introducción

La rinitis alérgica causa signos y síntomas parecidos al resfrío, como secreción nasal, picazón en los ojos, congestión, estornudos y presión en los senos nasales. No obstante, a diferencia de un resfriado, la rinitis alérgica no se produce por un virus, sino que se produce por una respuesta alérgica a los alérgenos en interiores o al aire libre, como el polen presente en árboles y gramíneas, esporas de hongos, los ácaros del polvo o pequeñas manchas de piel y saliva que arrojan los gatos, perros y otros animales con pelos o plumas.

La disciplina denominada Aeropalinología tiene como objetivo investigar el contenido de los granos de polen y esporas de hongos en la atmósfera y la relación con los factores que condicionan su producción y transporte. Los estudios que surgen de esta disciplina tienen diferentes aplicaciones, pero es en el área de las alergias, quizás, la más difundida y desarrollada, ya que estos granos de polen y esporas de hongos pueden entrar en contacto con los ojos y la mucosa nasal. Es sabido que se cuentan con altos niveles de alergias y asma en la población infantil, provocando además del malestar ya conocido, efectos en las actividades diarias como ausencia en sus estudios, disminución de la productividad, así como costos asociados al tratamiento de los síntomas.

Hasta el momento de comenzado el proyecto, no existía en Uruguay una forma de relacionar los datos aerobiológicos de un paciente mientras se le hace el tratamiento para su alergia. Si bien en Uruguay se estableció un rango de nivel aceptado para el polen y las esporas de hongos, de modo de saber si estos niveles son altos, medios o bajos, no existía una forma de difundirlos públicamente a la población. Siendo esta información de ayuda tanto para el usuario que sufre la alergia, como para la población de médicos que puede hacer uso de los mismos de manera de guiar el tratamiento.

Luego de una extensa investigación sobre trabajos relacionados en esta área, encontramos un gran número de aplicaciones que buscan ayudar a los pacientes con rinitis alérgica de diferente formas, las mismas se encuentran descriptas en el Capítulo 2. Todas esta aplicaciones contaban con limitantes para el usuario Uruguayo, algunas no estaban disponible para nuestro país, otras eran pagas, no estaban disponibles para alguno de los sistemas operativos más usados, no contaban con un

diario de síntomas, y principalmente, ninguna de ellas contaba con datos precisos para Uruguay. Así es que nace la necesidad de MIA, con la idea de informar y acercar a la población uruguaya más datos de los que ya conocen sobre sus alergias, y poner a disposición la medición de los pólenes y esporas de hongos que se realizan diariamente por parte de las palinólogas, datos que no se estaban difundiendo de forma pública y gratuita hasta el lanzamiento del proyecto. Así es que se decide crear una aplicación móvil para los sistemas operativos iOS y Android, que cuente principalmente con una forma amigable de mostrar las mediciones aeropalinológicas con los colores del semáforo, con un diseño y nomenclatura de fácil lectura para cualquiera sea la edad del usuario final, y además, contar con un diario de síntomas donde se puedan agregar eventos si en algún momento el usuario se sintió mal durante el día, de modo que pueda tener un registro no solo para si mismo, sino también para poder mostrarle a su médico tratante el día de la consulta. En un principio se tenía también la idea de mostrar en un mapa el arbolado presente en la ciudad de Montevideo, de manera que el usuario pudiera decidir su camino a sus diferentes actividades según la concentración de polen y esporas de hongos para ese día en particular, funcionalidad que finalmente no se pudo llevar a cabo por temas de tiempo y priorización, pero se sigue teniendo en cuenta para ser agregada en el futuro.

Una de las aplicaciones que ayudó e inspiró a este proyecto fue la llamada MASK-Air, diseñada por médicos alergólogos, con objetivo final aquellos pacientes que padecen rinitis alérgica, contiene también un diario de síntomas donde sus usuarios ingresan por 7 días seguidos cómo se han sentido respecto a su alergia, de donde las médicas tomaron ideas para realizar las preguntas que se harían luego en el diario de síntomas de MIA.

Antes de comenzar a implementar la aplicación se necesitó de un aproximado de cuatro meses de reuniones para la recolección de casos de usos, diseños de baja, media y alta fidelidad, y refinar requerimientos con los diferentes equipos de manera de bajar a tierra la idea que se tenía en un principio. Una vez obtenidos estos requerimientos necesarios para la publicación de la primer versión, y dado que se contaba con una fecha límite debido a la apertura de la policlínica de rinitis alérgicas por partes de las médicas en Noviembre del 2020, se crearon los casos de usos necesarios y procedimos a priorizar cada requerimiento en tres niveles diferentes: prioridad alta, media y baja, y se creó un cronograma que aseguraría la implementación de aquellos con prioridad alta y media y sus respectivas pruebas, dejando un margen de posibilidad para incluir también aquellos de prioridad baja.

Es así que surgen:

7 requerimientos de alta prioridad

- Registro de usuario
- Inicio de sesión
- Datos de usuario

- Datos de síntomas
- Obtener reporte
- Listar síntomas y toma medicamentos
- Agregar síntoma

4 requerimientos de media prioridad

- Datos de alergias
- Datos de medicamentos (utilizados en el tratamiento)
- Obtener detalle reporte
- Agregar Medicamento

5 requerimientos de prioridad baja

- Cambiar contraseña
- Ver/Editar perfil de usuario
- Agregar/Editar alergias
- Mostrar mapa
- Alarmas

Algunos de ellos cambiarían luego su prioridad durante la implementación, tema que se tratará en profundidad en el Capítulo 3.

La aplicación se creó utilizando el framework de JavaScript llamado React Native, tecnología que permite con un mismo código crear una aplicación para ambos sistemas operativos, iOS y Android, dándonos la posibilidad de alcanzar un mayor número de usuarios en el mismo tiempo de desarrollo, además se utilizo el framework de Python llamado Django para la implementación del backend, ya que fue creado específicamente con el propósito de la construcción de aplicaciones, por lo que facilita el trabajo a la hora de crear tanto aplicaciones web como móviles. Además el backend publica una API que es consumida por el frontend, para la creación de la misma, se utiliza la librería llamada Django REST Framework, que sigue las buenas prácticas y el diseño de una API REST. Durante el proyecto se vio la necesidad de crear una web donde las palinólogas puedan subir los reportes diarios de medición de el nivel de polen y esporas de hongos en el aire, para esto se hizo uso de una funcionalidad provista por Django, donde fácilmente se puede crear un administrador web para mostrar los datos almacenados en la base de datos, además de poder manipularlos, dando la posibilidad de agregar nuevas instancias de un modelo, y

eliminar o editar las existentes. Para la creación de la base de datos se utilizó la base datos relacional PostgresSQL, sumándole una extensión llamada PostGIS, que da la posibilidad de trabajar y manipular fácilmente datos geográficos, que nos sería de utilidad para guardar la ubicación de los árboles que se verían en el mapa de la aplicación móvil. Surge también la necesidad de crear una página web que presente el proyecto, mostrar información sobre el mismo, incluir enlaces a la descarga de la aplicación, presentar el equipo y ofrecer un contacto. Esta página web fue creada con el framework de JavaScript llamado Angular, por lo que esta página web contiene información estática sobre el proyecto. En el Capítulo 4 se detalla la arquitectura y el modelo relacional a los que se llego en base a cada uno de los módulos mencionadas anteriormente y como se relacionan.

Para la publicación de la API y la página web se utilizó un servidor provisto por la Facultad de Ciencias, punto que nos parecía importante a tener en cuenta, dado que al estar almacenando datos que pueden ser sensibles para el usuario, es pertinente mantenerlos dentro de Uruguay, por lo que le dimos importancia a buscar un servidor que mantuviera estos datos dentro del país. Para facilitar el manejo de esta publicación, se encapsuló la aplicación backend dentro de un contenedor, usando la herramienta Docker. Todas estas tecnologías y herramientas serán extendidas en el Capítulo 6.

También se hizo énfasis en la importancia de implementar pruebas automatizados del lado del backend, teniendo en cuenta que nuevos cambios siempre pueden introducir errores, por lo que sería engorroso probar todas las funcionalidades cada vez que se agregan nuevos cambios, se intentó tener un alto cubrimiento de todos los servicios publicados por la API, realizando un extenso número de pruebas automatizadas que simulan el consumo del lado del frontend con datos aleatorios y con resultados esperados. Estas pruebas son realizadas cada vez que se quiere agregar nuevo código al que ya está implementado, alertando siempre que se produzca algún error o el resultado no sea el que se registró como esperado. Vea el Capítulo 7 por más detalle sobre las pruebas y validaciones realizadas.

Como resultado se obtuvo una aplicación móvil funcional en el tiempo pautado, con los requerimientos de alta, media y algunos de baja prioridad, disponibles para ambas tiendas, un administrador el cual permite agregar los nuevos reportes de polen y esporas de hongo y una página web en la que se encuentran todas los datos sobre la aplicación y el proyecto. En un principio con quizás un bajo número usuarios, dado que la policlínica se vio afectada por la situación de pandemia del momento. Aún así, este número de usuarios se vio enormemente favorecido gracias a los medios de comunicación que se logró establecer contacto, brindando entrevistas para conocidos diarios, plataformas web y programas de televisión, fue así que llegamos a más de 2700 usuarios registrados en la base de datos, de los cuales cerca de 2000 tienen confirmada su cuenta a través del correo electrónico. Podrá consultar todo lo referente a las funcionalidades implementadas tanto en la aplicación, como en el administrador o en la página web en el Capítulo 5.

Capítulo 2

Estudio de trabajos relacionados

Desde el comienzo del proyecto el objetivo principal fue elaborar un Producto Mínimo Viable (MVP) de plataforma para ayudar a los pacientes con rinitis alérgica o hiperreactividad respiratoria, en el seguimiento de su condición para identificar los desencadenantes de sus alergias y colaborar en el tratamiento adecuado. Para cumplir con esto, por un lado, se pretendía compartir con el usuario los datos recolectados por las palinólogas que medirían el nivel de polen en el aire para los diferentes grupos polínicos, y se mostraría en una especie de semáforo al usuario con niveles de cuán crítico podría ser para sus alergias. Por otro lado, se tenía también, la idea de cruzar estos reportes de pólenes con datos geográficos, trabajando así con un equipo de la intendencia que nos proveería de la ubicación de los tipos arbóreos distribuidos en todo Montevideo, por lo que podríamos identificar entradas de los usuarios en donde hayan tenido un ataque de alergia, junto con los árboles y niveles de pólenes que se encontraban al momento del ataque.

A partir de las primeras reuniones con todo el equipo interdisciplinario, resultó necesario buscar plataformas con objetivos similares al de este proyecto. Por lo que se buscaron aplicaciones, sistemas o investigaciones que realizarán un trabajo parecido a lo que necesitábamos, haciendo uso de algunas o todas estas componentes: datos geográficos y síntomas de hiperreactividad respiratoria, fue así que logramos analizar nueve aplicaciones que nos servirían para sacar conclusiones antes de comenzar con el relevamiento de requisitos: Asthma Health App, Air Quality App, Propeller, DailyBreath, Asthmatic, FindAir, AsthmaMD, AsthmaTrack, MASK-Air. En este capítulo se desciben en detalle las aplicaciones relevadas.

2.1. MASK-Air

Figura 2.1: Aplicación MASK-Air

Mask-air (`https://www.mask-air.com/`) es una aplicación para pacientes que sufren de rinitis alérgica, es gratuita y está disponible en varios idiomas, entre ellos español, y disponible tanto para plataformas iOS como Android, sin embargo no está disponible para Uruguay. Está aplicación ha sido desarrollada por alergólogos y ha sido validada científicamente. Cuenta con un plan de seguimiento de 7 días, donde invitan a sus usuarios a registrar sus síntomas por al menos 7 días seguidos, teniendo que contestar algunas preguntas, guardar su tratamiento actual y monitorear las consecuencias en su calidad de vida.

Contiene interacción con un proveedor del cuidado de la salud, y aseguran que los datos ingresados por los usuarios ayudan a estos proveedores a ofrecer una estrategia apropiada de tratamiento para la rinitis alérgica para atravesar el año a través de las diferentes estaciones meteorológicas. Mask-air también se nutre de toda la información provista por sus usuarios para compartirla con investigadores, que en el mediano plazo, aseguran, serán capaces de beneficiar a la comunidad gracias a la colaboración de la aplicación. Investigadores serán capaces de entender mejor la efectividad de los tratamientos, derivando en un mejor manejo de la rinitis en el futuro.

Fue en esta aplicación en la que luego se basaron las doctoras del equipo para determinar qué preguntas hacer a los usuarios y posibles respuestas, dado que ha sido desarrollada por un equipo internacional formado por los alergólogos más importantes del mundo y dirigido por el Prof. Jean Bousquet, un famoso investigador francés, reconocido por su producción científica.

2.2. Asthma Health App (AHA)

Asthma Health App [3] es una aplicación móvil creada en EEUU que ya no se encuentra disponible en las tiendas, se usó durante los años 2015 y 2016 para recolectar información de los síntomas del usuario y relacionarlo con la calidad del aire en la que se encontraba. Para esto, la aplicación recopilaba a cada hora, la latitud y longitud en la que se encontraba el usuario, haciendo uso del GPS del móvil. Este proyecto se realizó con la idea de validar la viabilidad del uso de dispositivos

inteligentes para ayudar al manejo de enfermedades crónicas e investigaciones de la mismas, se alcanzó 6346 usuarios que estuvieron de acuerdo en compartir sus datos para la investigación.

Para el estudio, se creó una aplicación para iPhone dirigida a pacientes con asma, utilizando el framework ResearchKit de código abierto, provisto por Apple. Esta aplicación invitaba a sus usuarios a utilizarla como un diario electrónico en donde mantener un registro de sus síntomas y posibles desencadenantes. También ofrecía recordatorios para los medicamentos, vídeos informativos, además de información sobre el clima y la contaminación, y de esta manera relacionarlos con los síntomas que el usuario reportaba.

Usuarios

Los primeros pasos al descargar la aplicación eran llenar un formulario sobre si se era mayor de edad, embarazada, y alfabetización en el idioma. Luego se le hacía saber sobre el estudio, sus beneficios y riesgos, opciones de compartir sus datos y protección de la privacidad, mediante una serie de pantallas informativas. Luego se le hacían preguntas para comprobar que realmente hubiese entendido sobre el proyecto, y finalmente se lo hacía firmar electrónicamente. Luego de firmado, se pedía un mail para verificación y se mandaba el formulario llenado anteriormente con la firma electrónica.

Las estadísticas mostraron que la AHA fue descargada por 48054 usuarios, de los cuales solo 11214 fueron consentidos en la primer fase de preguntas, de ellos, 10010 verificaron su mail, de los cuales:

- 6346 compartieron ampliamente sus datos
- 2053 compartieron poco
- 1611 se retiraron

Cabe destacar que se les dio la opción de compartir sus datos solamente con el equipo de estudio y sus socios, o compartir también con investigadores calificados de todo el mundo. Se les dio la opción también, de abandonar el estudio en cualquier momento.

Recolección de datos

Luego del usuario ya haber iniciado sesión, en los primeros 3 días se le ingresaron ciertas encuestas en una sección de la aplicación, que debía llenar para obtener información, entre otras cosas, de:

- Antecedentes del asma, incluyendo frecuencia y tiempo de los síntomas, y cuánto los limitaba en sus actividades diarias.
- Experiencia con el asma, incluyendo desencadenamientos y como se manejaban personalmente.
- Historial médico, es decir alergias y medicamentos actuales y posteriores.

- Demografía (etnia, raza, edad, género, ingresos y educación)

Ninguna de las preguntas tenían una respuesta por defecto para evitar sesgos, y se daba la opción de omitir la pregunta en caso de que así se quisiera.

También se recolectó información de la ubicación del usuario mediante el uso del GPS del móvil, vinculándolo con los informes de la calidad del aire. En una primera versión, se enviaba la calidad del aire en el momento en que el cliente entraba a la aplicación, pero dado que no se tenía suficiente precisión, en una segunda versión se optó por obtener cada hora la latitud y longitud, si la ubicación cambiaba.

Limitaciones

Dado que fue una aplicación desarrollada solamente para iPhone, se identifica un sesgo socioeconómico en el estudio. Un estudio de investigación de Pew en 2016, revela que los propietarios de iPhone cuentan con niveles de educación e ingreso mayores a otros modelos de teléfonos, por lo que se entiende que desarrollar la aplicación también para dispositivos que usen Android, podría alcanzar grupos más diversos de la población.

Se notó una disminución de la actividad por persona a lo largo del estudio, se cree que puede estar ligado a optar por el uso de otras aplicaciones que provean entretenimiento, por lo que se aconseja desarrollar esta aplicación con principios psicosociales en mente, y no enfocarse solamente en lo técnico, de manera de retener a los usuarios a lo largo del tiempo.

2.3. Air Quality App - BreezoMeter

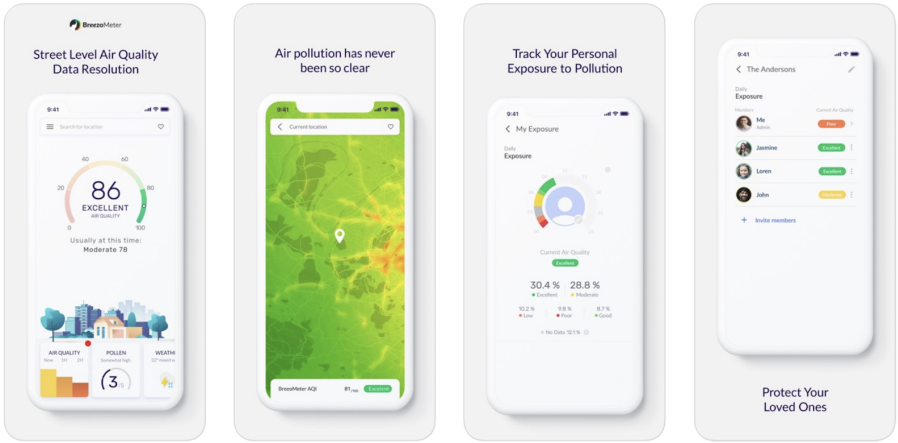


Figura 2.2: Aplicación Air Quality App - BreezoMeter

Air Quality App (`https://breezometer.com/`) fue creada en 2012 por Ran Korber dada la necesidad de encontrar un lugar donde vivir libre de contaminación, debido a familiares con salud sensible (incluido asma), sorprendido de no encontrar una solución a su necesidad, decidió crear Air Quality App.

Busca ofrecer a sus usuarios la posibilidad de disminuir su exposición a la contaminación del aire, polen y alertas de incendios. Les ofrece un pronóstico de la calidad del aire en las próximas 6 horas, y la posibilidad de observar el estado del mismo en tiempo real en un mapa de calor.

Utilizan infraestructura de big data para recopilar la calidad del aire, polen y mediciones climáticas, extraídas de miles de fuentes, entre ellas, más de 7000 estaciones oficiales de monitoreo de calidad del aire alrededor de todo el mundo. Cuentan con un algoritmo propio de dispersión, que calcula la calidad del aire a cada hora, que los ayuda a entender cómo se mueve y dispersa, lo cual les permite ofrecer datos precisos y en tiempo real.

Se aseguran de proteger la información del usuario, no se venden ni comparten sus datos con librerías de terceros. Solo se almacena externamente la información de los lugares favoritos que el usuario eligió, para poder enviarle notificaciones cuando cambia la calidad del clima de los mismos. Además, se utiliza la localización del mismo para calcular la calidad del aire a su alrededor, su exposición y enviar alertas.

2.4. Propeller

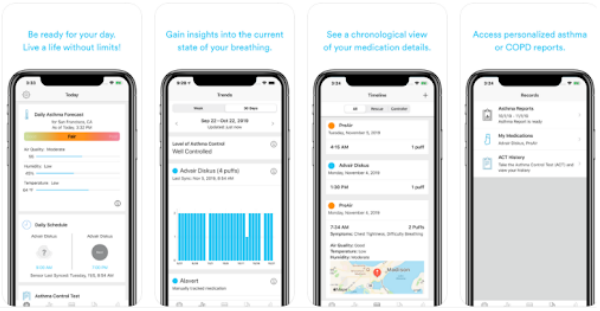


Figura 2.3: Aplicación Propeller

Propeller Health (`https://www.propellerhealth.com/`) desarrolla productos que ayudan a las personas con asma o EPOC a controlar su afección en colaboración con su proveedor de atención médica. La plataforma Propeller es utilizada por pacientes, proveedores y organizaciones de atención médica en EE.UU, Europa y Asia. Los usuarios pueden acceder a esta plataforma mediante sus teléfonos inteligentes o computadores.

Funcionamiento

Está aplicación funciona en conjunto con un sensor que se conecta a los inhaladores que los paciente ya tienen. Los sensores rastrean automáticamente dónde, cuándo y con qué frecuencia los pacientes usan su medicamento, y envían esta información a la aplicación. La recolección de información permite aprender sobre los brotes y el uso de medicamentos, para luego generar ideas y recordatorios personalizados para ayudar a los pacientes en su tratamiento.

El sistema también brinda la posibilidad de personalizar un cronograma de medicamentos para que los pacientes puedan recordar tomarlo a tiempo. Podrán verificar su pronóstico diario de asma, incluidos los detalles de calidad del aire, humedad y temperatura.

Con el consentimiento del paciente, esta aplicación puede comunicarse con su médico para mostrar cómo le está yendo en su tratamiento y compartir información sobre el manejo de su enfermedad y medicación.

2.5. DailyBreath

DailyBreath es una aplicación que está en funcionamiento solamente en Estados Unidos, pensada para ayudar a usuarios que sufren de asma o alergias a aprender sobre su propia enfermedad, sus síntomas y sus desencadenantes. Parte de la base de que la exposición a la contaminación y el polen convergen con el clima, para desencadenar en los síntomas o brotes del paciente.

La aplicación se nutre de las experiencias que desencadenan los síntomas de los usuarios, una de sus funcionalidad principales es el proveer de una índice de riesgo personalizado para cada usuario a lo largo del tiempo. Cuando el usuario es nuevo, ya que no se cuenta con muchas experiencias del mismo, se ofrece un mapa de sus últimos 5 brotes. Buscan que el usuario sepa exactamente, ante un brote, cuándo, dónde, y bajo qué condiciones de clima sucedieron.

Además ofrece recomendaciones diarias para prevención, rastreo personal de brotes, y un mapa comunitario que cruza los síntomas de las experiencias de todos los usuarios de la aplicación.

2.6. Asthmatic

Asthmatic es una aplicación hecha para pacientes con asma, que predice hasta con 5 días de anticipación, disponible mundialmente pero únicamente para usuarios de Apple.

Funcionamiento

Asthmatic calcula la calidad del aire en función de una gran variedad de condiciones climáticas como la temperatura, la humedad, la contaminación y los niveles de ozono, para generar una puntuación que va de mala a excelente. Los pacientes podrán ver un pronóstico por día en el que se les indicará qué condiciones son malas y se les aconsejará cómo lidiar con ellas, además permite agregar o eliminar ubicaciones.

Está información fue recavada del App Store, no fue encontrada una página web de la organización. Por otro lado se sabe que el lenguaje de la misma es en ingles y que tiene un costo de 2.99 dólares.

2.7. FindAir – Asthma Diary

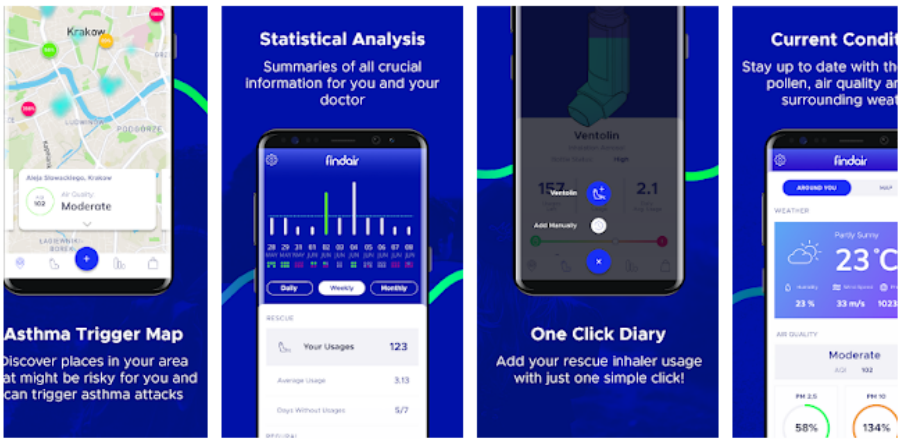


Figura 2.4: Aplicación FindAir – Asthma Diary

FindAir (`https://findair.eu/`) en un sistema de inteligencia artificial para pronosticar y prevenir las exacerbaciones del asma, disponibles para los usuarios de Android. Esta aplicación utiliza los datos recopilados por el sistema, que son procesados por muchos métodos en el campo de aprendizaje automático y el análisis estadístico. El propósito del sistema es identificar el factor principal que causa los ataques de asma en los pacientes y predecir en tiempo real las exacerbaciones del asma y la EPOC.

Funcionamiento

Para hacer uso de la aplicación es necesario que los usuarios compren un sensor inteligente que se deberá colocar en el inhalador del paciente. Con el sensor se recopilan fácilmente los datos sobre cada uso del inhalador, sin necesidad de completar datos manualmente. Los datos obtenidos, entre ellos son, la ubicación del usuario, cuándo utilizó la medicación, y cruzados con datos de información ambiental como la contaminación del aire, las condiciones climáticas y los alérgenos en el área, con el fin de generar informes para los pacientes y sus médicos y recibir en tiempo real notificaciones de los peligros en el medio ambiente. El sistema también brinda un sistema de alertas y de recordatorios para no olvidar tomar la medicación.

2.8. AsthmaMD

AsthmaMD (`https://www.asthmamd.org/`) es una aplicación gratuita que ofrece una especie de diario a sus usuarios, donde pueden agregar sus síntomas, sus causas y medicamentos, teniendo la opción de compartir estos datos y gráficas con sus médicos para que se incluyan en sus registros médicos.

Este proyecto reúne la información de todos sus usuarios de manera anónima, para ayudar a investigadores sobre las causas del asma y sus relaciones externas. Esta opción es optativa para los usuarios, y en caso de estar de acuerdo, se estará

compartido sus datos de manera encriptada y anónima sobre: la gravedad de su asma, ataques, desencadenantes, hora, día, y ubicación. De esta forma los investigadores pueden tener visibilidad de la correlación entre el nivel de asma que tiene un barrio, día y hora, contaminación y clima, entre muchas otras.

Algunas otras características interesantes que ofrece esta aplicación son:

- Agregar y revisar anteriores ataques de asma
- Visualizar un resumen de los ataques de manera gráfica, indicando nivel de gravedad
- Presentar un plan de acción y guiar al usuario a través del mismo
- Agregar alertas para tomar sus medicamentos

2.9. AsthmaTrack

Es un a aplicación paga disponible únicamente para usuarios de Apple, permite registrar inhalaciones, medicamentos y síntomas.

Cuenta con una sección denominada *Diario*, donde el usuario puede agregar entradas en él, sobre medicamentos que tomó, síntomas, disparos de inhaladores, etc. En particular, al ingresar una entrada de síntomas, se le presenta un formulario con opciones, alguna de ellas son:

- Jadeo
- Tos
- Opresión en el pecho
- Falta de aire
- Rapidez del latido del corazón
- Fiebre/Transpiración
- Labios pálidos
- Insomnio

Los usuarios pueden seleccionar opciones entre: ninguno, ligero, medio y fuerte.

En otra sección, denominada herramientas, y con los datos recopilados, el sistema crea gráficos. Todo la información recopilada y generada por el sistema puede ser compartida por correo electrónico a su médico o exportada en formato csv. Además, se aseguran que las entradas puedan ser agregadas sin necesidad de internet de manera de hacerlo en el momento en el que sucede.

2.10. Conclusión

Por distintas razones, ninguna de estas aplicaciones contaba exactamente con el alcance que buscábamos en nuestro proyecto, ya sea porque no están disponibles para Uruguay, no son gratuitas, no están disponibles en español, no se encuentran disponibles para todas las plataformas móviles, o no cumplen con todos los requisitos requeridos en el proyecto.

Luego de este análisis, destacamos muy buenos componentes e ideas que nos sirvieron para el desarrollo de nuestra propia plataforma, entre los cuales se encuentran:

- Asegurar ser transparentes ante el usuario en cómo se comparten y utilizan sus datos, es necesario mostrar una serie de pantallas en donde se cuente brevemente sobre el proyecto, y se detalle cómo se estarán usando sus datos. Es imprescindible contar con la confirmación del usuario, por lo que deben estar estas pantallas al momento de registrarlos, y únicamente seguir con el proceso si el usuario está de acuerdo luego de leer los términos.

- Dado que los usuarios no necesariamente son idóneos en medicina, es necesario guiarlos si se quiere obtener mayor información, si se quiere saber sobre algún síntoma, entonces el mejor enfoque es presentarles un formulario con síntomas predeterminados por médicos, que son los que mejor saben cuáles pueden haber experimentando, pero que quizás los usuarios no se les ocurriría si solamente se les presenta un campo abierto donde escribir síntomas. También como se mencionó anteriormente, no utilizar respuesta por defecto en las opciones, para evitar el sesgo de los datos.

- Los brotes y síntomas se repiten para un usuario, según condiciones del aire, estación del año y clima. Por lo que la aplicación también sirve, para ellos, como una forma de aprender sobre sí mismos, por lo que es importante que cuenten con una sección donde puedan ver sobre sus propios reportes anteriores, y puedan reconocer patrones. No solo ayuda al doctor a saber con exactitud qué está causando sus síntomas, sino que ellos pueden ver gráficamente qué los afecta y de qué manera.

- Es de ayuda el uso de herramientas gráficas para visualizar la información de manera amigable. Si se quiere identificar zonas críticas, el uso de un mapa es una buena opción, diferenciando con colores las zonas según qué tan críticas sean. Si se quiere mostrar un resumen de los datos obtenidos del usuario, se puede optar por gráficas para observar los resultados generales.

Capítulo 3

Relevamiento de requerimientos

En este capítulo se describe el relevamiento de requerimientos para la aplicación móvil que estaba prevista originalmente, y el administrador y la página web que surgieron en el proceso de relevamiento.

3.1. Aplicación móvil

Como se mencionó en el Capítulo 2, se descartó la idea de reutilizar alguna aplicación ya existente, por tanto se realizaron algunas reuniones más con el grupo completo para relevar requerimientos. De estas reuniones se obtuvieron los requerimientos para desarrollar la aplicación y a partir de esto se construyeron prototipos usando la herramienta de media fidelidad Balsamiq. El prototipado consta de implementación parcial, la cual permite a los desarrolladores y usuarios entender mejor los requisitos, así como entender cuáles son necesarios y cuáles son deseables, teniendo como ventaja acotar el riesgo. Esta técnica parte de una base de requisitos, en nuestro caso aquellos que fueron recabados de las reuniones con el equipo. Los prototipos nos permitieron plasmar lo que se hablaba en las reuniones, convergiendo a un producto más tangible.

Una vez desarrollado el prototipo y habiendo hecho los cambios pertinentes sugeridos en las reuniones, se realizó un documento de casos de uso. Los casos de uso son una descripción de las actividades que pueden realizar los usuarios del sistema mediante el uso de la aplicación, buscando definir en líneas generales el comportamiento del sistema para validar los posibles escenarios y comportamientos que se espera tener. Es decir que describen cómo el sistema debe comportarse desde el punto de vista del usuario, es por esto que se clasifican a los casos de uso como requisitos funcionales, ya que se describe qué debe hacer el sistema, pero no cómo debe hacerlo.

Los casos de uso fueron divididos en tres categorías, estas fueron alta, media y baja prioridad. En la categoría alta prioridad se agruparon aquellos casos de uso que eran necesarios para el funcionamiento del sistema y que se quería que fueran los primeros en desarrollarse para poder tener una retroalimentación lo más temprano posible. Bajo la categoría prioridad media, se agruparon aquellos casos de uso que se desarrollaron en una segunda etapa, casos de uso no tan críticos pero que sí era

deseados en el sistema. Y por último, bajo la categoría prioridad baja, se encontraron los casos de uso que no eran esenciales para el sistemas, algunos de los cuales aún no se tenía la información necesaria para llevarlos a cabo, estos casos de uso se implementarían solo si el tiempo de hacerlo en el plazo determinado era suficiente.

A continuación se muestran las distintos bosquejos plasmados en la herramienta de media fidelidad, seguido de su correspondiente caso de uso. Los mismos están agrupados según su prioridad.

3.1.1. Prioridad Alta

1. **Registro de Usuario**

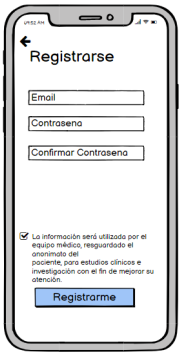


Figura 3.1: Balsamiq Prototipo: Registro de Usuario

Nombre	**Registrarse**
Actor	Paciente
Descripción	Comienza cuando el usuario quiere crear una nueva cuenta. Se le solicita el email, contraseña, confirmación de la misma y aceptar las condiciones de privacidad. El botón de registrarse sólo se habilitará si llenó todos los campos y las condiciones están aceptadas. Dadas estas condiciones, el usuario puede registrarse, y si las contraseñas coinciden, se agrega el nuevo usuario satisfactoriamente, sino se le muestra un mensaje de error.

Cuadro 3.1: Caso de uso - Registrarse

2. **Inicio de Sesión**

Figura 3.2: Balsamiq Prototipo: Inicio de Sesión

Nombre	**Iniciar sesión**
Actor	Paciente
Descripción	Comienza cuando el usuario desea ingresar a la aplicación. Se solicita el mail y la contraseña. Si el mail no existe, le aparecerá un mensaje que debe registrarse, si el email existe y la contraseña es la correcta, la sesión queda iniciada.

Cuadro 3.2: Caso de uso - Iniciar Sesión

3. **Datos de usuario**

Figura 3.3: Balsamiq Prototipo: Datos de usuario

Nombre	**Datos del paciente**
Actor	Paciente
Descripción	La primera vez que el usuario inicia sesión, se le harán preguntas sobre sus datos: ▪ Nombre* ▪ Fecha de nacimiento* ▪ Género* ▪ Foto de perfil Estos datos serán guardados en el sistema para posterior consulta.

Cuadro 3.3: Caso de uso - Datos del paciente

4. **Datos de síntomas**

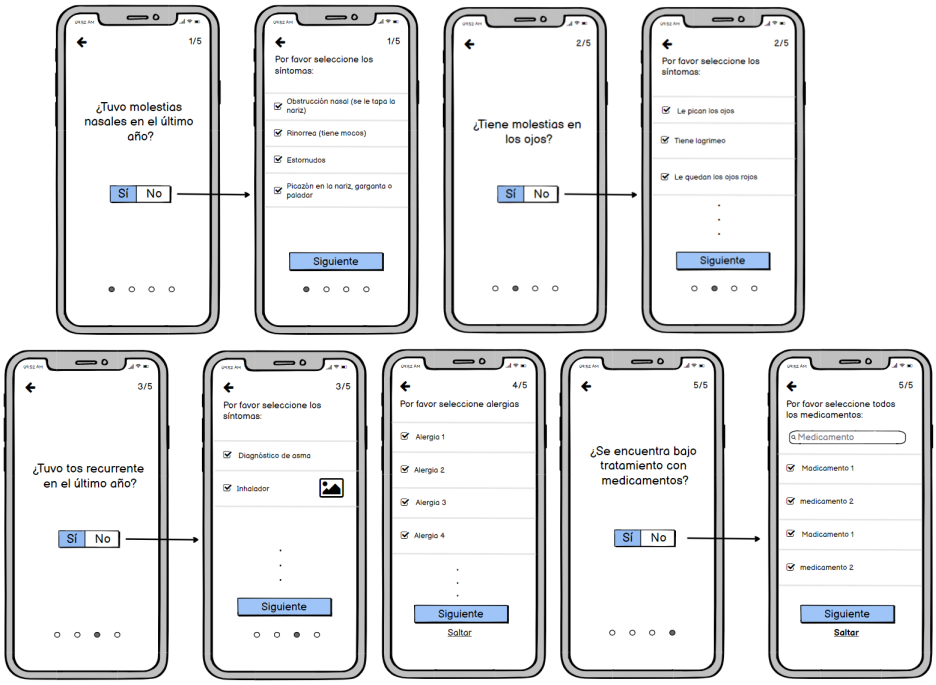


Figura 3.4: Balsamiq Prototipo: Datos de síntomas

Nombre	**Datos de síntomas**
Actor	Paciente
Descripción	La primera vez que el usuario inicia sesión, se le harán preguntas sobre sus síntomas y podrá seleccionar todas las opciones que necesite: **¿Tuvo molestias nasales en el último año?** Si la respuesta es afirmativa: ▪ Obstrucción nasal (se le tapa la nariz) ▪ Rinorrea (tiene mocos) ▪ Estornudos ▪ Picazón en la nariz, garganta o paladar **¿Tiene molestias en los ojos?** Si la respuesta es afirmativa: ▪ Le pican los ojos ▪ Tiene lagrimeo ▪ Le quedan los ojos rojos **¿Tuvo tos recurrente en el último año?** Si la respuesta es afirmativa: ▪ Diagnóstico de asma ▪ Inhalador (con foto/icono descriptivo) Estos datos serán guardados en el sistema para posterior consulta.

Cuadro 3.4: Caso de uso - Datos de síntomas

5. **Obtener reporte**

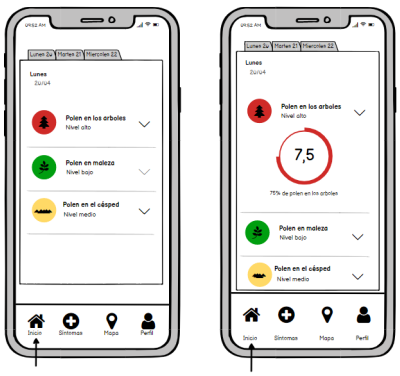


Figura 3.5: Balsamiq Prototipo: Obtener reporte

Nombre	**Obtener reporte**
Actor	Paciente
Descripción	Comienza cuando el usuario inicia sesión, se desea traer los datos diarios o semanales del polen y esporas de hongos en el aire y esporas de hongos, estos datos incluyen : ▪ **nivel de polen** en los árboles ▪ **nivel de polen** en hierbas ▪ **nivel de esporas** en hongo ▪ Alerta personalizada en base a las alergias del paciente cuando el riesgo sea alto. Los niveles de polen o esporas serán: **Alto, Medio, Bajo**

Cuadro 3.5: Caso de uso - Obtener reporte

6. **Listar síntomas y toma medicamentos**

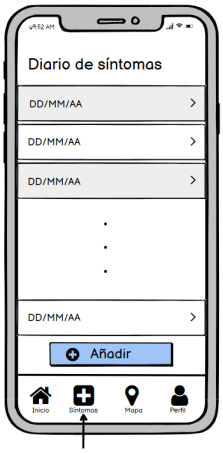


Figura 3.6: Balsamiq Prototipo: Listar síntomas y toma medicamentos

Nombre	**Listar síntomas y toma medicamentos**
Actor	Paciente
Descripción	Comienza cuando el usuario selecciona la opción del menú síntomas y podrá ver sus medicamentos y síntomas ingresados, de forma descendente por la fecha en que se registraron los mismos.

Cuadro 3.6: Caso de uso - Listar síntomas y toma medicamentos

7. **Agregar síntoma**

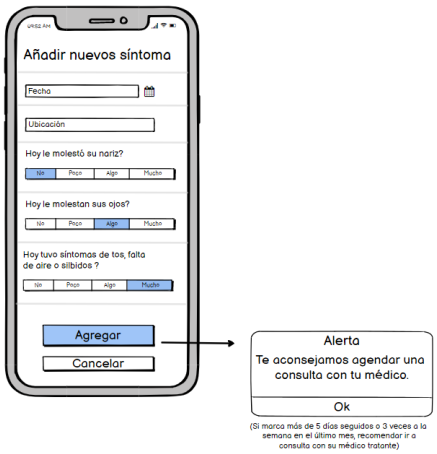


Figura 3.7: Balsamiq Prototipo: Agregar síntomas

Nombre	**Agregar síntoma**
Actor	Paciente
Descripción	Comienza cuando el usuario selecciona la opción agregar síntoma de la pantalla de síntomas, el sistema debe ser capaz de guardar un nuevo síntoma, para esto el usuario deberá proveer datos fecha y utilizaremos la ubicación actual por defecto, la cual podrá ser editada. Por otro lado se requerirán que el usuario conteste una serie de preguntas: ▪ ¿Hoy le molesto su nariz?* ▪ ¿Hoy le molestan los ojos?* ▪ ¿Hoy tuvo síntomas de tos, falta de aire o silbidos?* ▪ ¿Hoy se despertó en la noche o antes del despertador en la mañana?* ▪ ¿Faltó al trabajo o a sus estudios? ▪ ¿Está utilizando los medicamentos indicados? A lo que el usuario deberá responder con No/Poco/Algo/Mucho o Si/No/A veces. Y contará con un área de texto para que el usuario agregue comentarios que considere pertinentes. Una vez que el síntoma se guarda en el sistema, este deberá corroborar si el usuario tuvo otros síntomas frecuentemente, si marca más de 7 días seguidos o 3 veces a la semana en el último mes, se le mostrará una alerta que recomienda ir a consulta con su médico tratante. Trataremos estos datos como variables.

Cuadro 3.7: Caso de uso - Agregar síntoma

3.1.2. Prioridad Media

1. **Datos de alergias**

Figura 3.8: Balsamiq Prototipo: Datos de alergias

Nombre	**Datos de alergias**
Actor	Paciente
Descripción	La primera vez que el usuario inicia sesión, se le solicitará seleccionar de una lista de alergias aquellas que el usuario padezca.

Cuadro 3.8: Caso de uso - Datos de alergias

2. **Medicamentos**

Figura 3.9: Balsamiq Prototipo: Datos de medicamentos

Nombre	**Datos de medicamentos (utilizados en el tratamiento)**
Actor	Paciente/Médico
Descripción	La primera vez que el usuario inicia sesión, se le solicitará seleccionar de una lista de medicamentos aquellos que el usuario utiliza actualmente, podrá ser con ayuda del médico tratante.

Cuadro 3.9: Caso de uso - Medicamentos

3. **Reporte de Polen**

Nombre	**Obtener detalle reporte**
Actor	Paciente
Descripción	Comienza cuando el paciente desea obtener más información de la concentración de en uno de los tipos polínicos o fúngicos, esta acción traerá: ▪ Granos/cm3 o esporas/cm3 dependiendo de si el grupo consultado es polínicos o fúngicos ▪ Datos específicos de cada una de los elementos que conforman el conjunto, detallando su nivel de polen en alto, medio, bajo.

Cuadro 3.10: Caso de uso - Reporte de Polen

4. **Agregar Medicamento**

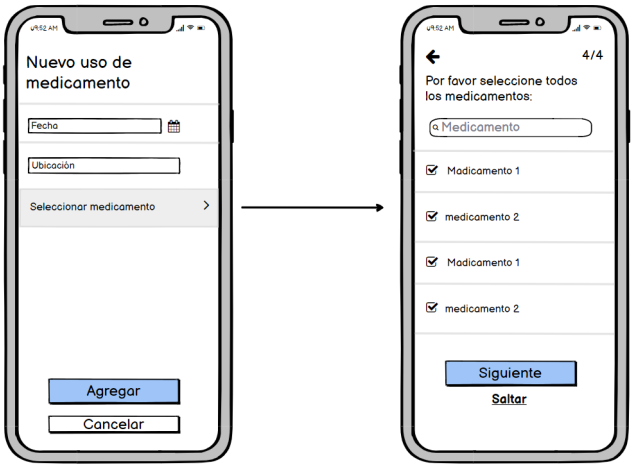


Figura 3.10: Balsamiq Prototipo: Agregar Medicamento

Nombre	**Agregar Medicamento**
Actor	Paciente
Descripción	Comienza cuando el usuario selecciona la opción agregar medicamento de la pantalla de síntomas, el sistema debe registrar el uso de medicamentos por fuera del tratamiento del paciente, para esto el usuario deberá proveer datos de dónde y cuándo se utilizó el medicamento, por defecto estos datos están precargados con fecha y ubicación actual, pero el usuarios podrá modificarlos. También se requerirá que el usuario indique el o los medicamentos utilizados, los mismos los deberá elegir de una lista ya definida en el sistema.

Cuadro 3.11: Caso de uso - Agregar Medicamento

3.1.3. Prioridad Baja

1. **Cambiar contraseña**

Figura 3.11: Balsamiq Prototipo: Cambiar contraseña

Nombre	**Cambiar contraseña**
Actor	Paciente
Descripción	Si el usuario no recuerda su contraseña, puede solicitar el cambio de la misma. Para ello se le solicita su email y la confirmación del mismo para enviarle un mail con un código. En la pantalla siguiente se le solicita el código enviado anteriormente, la nueva contraseña y la confirmación de la misma. Se contará con la opción de volver a enviar el código. Si las dos nuevas contraseñas coinciden, el cambio es realizado con éxito, sino, se le muestra un mensaje de error.

Cuadro 3.12: Caso de uso - Cambiar contraseña

2. **Perfil de usuario**

Figura 3.12: Balsamiq Prototipo: Perfil de usuario

Nombre	**Ver/Editar perfil de usuario**
Actor	Paciente
Descripción	Comienza cuando el paciente desea consultar sus datos personales, para esto el sistema deberá mostrar los siguientes datos del usuario actual: ▪ Nombre* ▪ Foto * ▪ Fecha de nacimiento* ▪ Género* ▪ Dirección* ▪ Alergias ▪ Cerrar sesión Los datos con * se podrán editar.

Cuadro 3.13: Caso de uso - Perfil de usuario

3. **Alergias del usuario**

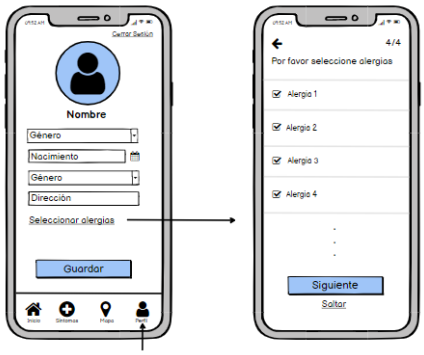


Figura 3.13: Balsamiq Prototipo: Agregar/Editar alergias

Nombre	**Agregar/Editar alergias**
Actor	Paciente/Médico
Descripción	Comienza cuando el usuario selecciona la opción agregar alergias desde el perfil, aquí el usuario podrá ingresar nuevas o editar alergias ya existentes. Una vez que el usuario da guardar, las alergias quedarán registradas en el sistema para posteriores consultas.

Cuadro 3.14: Caso de uso - Agregar/Editar alergias

4. **Mapa**

Figura 3.14: Balsamiq Prototipo: Mostrar mapa

Nombre	**Mostrar mapa**
Actor	Paciente
Descripción	Comienza cuando el usuario selecciona la opción mapa del menú, en esta pantalla se refleja con colores el grado de polen existente en el mapa.

Cuadro 3.15: Caso de uso - Mostrar mapa

5. **Alarmas**

Nombre	**Alarmas**
Actor	Paciente
Descripción	Se desplegarán notificaciones para recordarle al paciente la toma de los medicamentos. Falta definir los mensajes que se le mostrarán.

Cuadro 3.16: Caso de uso - Alarmas

3.2. Administrador

Dada la necesidad de que algunos datos sean consultados, creados, editados o eliminados por cualquier integrante del equipo de forma dinámica, es que se piensa en una página web que cumpla la función de administrador, en el que únicamente los usuarios con permisos especiales podrán iniciar sesión.

Las funcionalidades deseadas en el administrador son las descritas en los siguientes casos de uso.

Nombre	**Crear usuario administrador**
Actor	Equipo de Ingeniería
Descripción	Comienza cuando se quiere crear una nueva cuenta con permisos de super usuario que sea capaz de iniciar sesión de la página web del administrador, de forma de poder realizar cambios en el sistema.

Cuadro 3.17: Caso de uso - Crear usuario administrador

Nombre	**Iniciar sesión**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario desea ingresar al administrador. Se solicita el mail y la contraseña.

Cuadro 3.18: Caso de uso - Iniciar sesión

Nombre	**Cambiar Contraseña**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario desea cambiar su contraseña. Por razones de seguridad, se pide que se introduzca primero su contraseña antigua y luego se introduzca la nueva contraseña dos veces para verificar que se ha escrito correctamente.

Cuadro 3.19: Caso de uso - Cambiar Contraseña

Nombre	**Crear grupo polínico o fúngico**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario agrega un nuevo grupo polínico o fúngico. Para esto el usuario deberá ingresar, *id, nombre, unidad de medida, nivel alto y nivel medio*, de forma obligatoria, y de forma opcional podrá llenar los campos *Nivel alto sumatoria, Nivel medio sumatoria*. Por defecto se deben crear tres grupos polínicos o fúngicos mediante una migración, los mismos son Árboles, Hierbas y Hongo, que en principio serán los únicos grupos existentes.

Cuadro 3.20: Caso de uso - Crear grupo polínico o fúngico

Nombre	**Editar o consultar grupo polínico o fúngico**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere editar o ver el detalle de un grupo polínico o fúngico. El usuario podrá consultar los datos del mismo así como modificar cualquiera de los campos completados al crear el grupo polínico o fúngico

Cuadro 3.21: Caso de uso - Editar grupo polínico o fúngico

Nombre	**Listar grupos polínicos o fúngicos**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere ver los distintos grupos polínicos o fúngicos. Para esto se proporcionará una vista con una lista de los mismos, mostrando en ella los nombres de cada uno.

Cuadro 3.22: Caso de uso - Listar grupos polínicos o fúngicos

Nombre	**Crear tipo polínico o fúngico**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario agrega un nuevo grupo polínico o fúngico. Para esto el usuario deberá ingresar, *nombre común, grupo polínico, nivel alto y nivel medio, Picks presente*, de forma obligatoria, siendo el grupo polínico alguno de los grupos polínico o fúngico creados en el sistema, y de forma opcional podrá llenar los campos *tipo, nombre científico, familia, alergenicidad, nombre alergia, Archivo pdf*. Por defecto se crearán todos los tipos polínicos o fúngicos que al momento se sabe que producen alergias mediante una migración, si a futuro se descubren nuevos tiempos polínicos o fúngicos que perjudican la salud de los pacientes, entonces se los podrá ingresar.

Cuadro 3.23: Caso de uso - Crear tipo polínico o fúngico

Nombre	**Lista de los tipos polínicos o fúngicos**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere ver los distintos tipos polínicos o fúngicos. Para esto se proporcionaráuna vista con una lista de los mismos, mostrando en ella los nombres de cada uno.

Cuadro 3.24: Caso de uso - Listar grupos polínicos o fúngicos

Nombre	**Editar tipo polínico o fúngico**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere editar un tipo polínico o fúngico. El usuario podrá modificar cualquiera de los campos completados al crear el tipo polínico o fúngico.

Cuadro 3.25: Caso de uso - Editar tipo polínico o fúngico

De los casos de uso deseados para el administrador, solo aquellos que involucran a los reportes son de prioridad alta, y el resto son de prioridad baja.

Nombre	**Crear Reporte**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere agregar un nuevo reporte. Algunos de los campos a completar estarán cargados por defecto, estos son: ▪ Fecha: tendrá por defecto la fecha y hora actual y en caso de ser necesario podrá ser modificada. Este campo es obligatorio. ▪ Tiempo total: tendrá por defecto el valor 1440, y en caso de ser necesario podrá ser modificada. Este campo es obligatorio. ▪ Porcentaje muestreado: tendrá por defecto el valor 0.1, y en caso de ser necesario podrá ser modificado. Este campo es obligatorio. Se deberá completar el campo *tiempo extra* de forma obligatoria. Y además por cada tipo polínico o fúngico detectado en la muestra, se podrá ingresar su nivel de polen o espora de hongo muestreado.

Cuadro 3.26: Caso de uso - Crear Reporte

Nombre	**Editar Reporte**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere editar un reporte. El usuario podrá modificar cualquiera de los campos completados al crear el mismo.

Cuadro 3.27: Caso de uso - Editar Reporte

Nombre	**Listar reportes**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere ver todos los reportes. Para esto se listan los mismos, mostrando los atributos *Fecha, Tiempo real, Volumen de polen, Volumen en esporas de hongos*, a su vez se quiere que la lista esté ordenada descendente por fecha.

Cuadro 3.28: Caso de uso - Listar reportes

Nombre	**Descargar reporte**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere descargarse un archivo que contenga la información de un reporte. En la lista de reportes o en el detalle de cada uno de ellos, se podrá descargar un archivo en formato compatible con Excel.

Cuadro 3.29: Caso de uso - Descargar reporte

Nombre	**Agregar Información**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere agregar un nuevo texto con información del proyecto. Se quiere tener la posibilidad de agregar textos sobre el proyecto, que puedan ser modificados por cualquier integrante del equipo y que sea consumido por la aplicación o página web. En principio esto se usará para alojar los términos y condiciones de la aplicación, pero se podría crear cualquier otro texto con el fin de mostrarlo en los distintos sitios.

Cuadro 3.30: Caso de uso - Agregar Información

Nombre	**Editar Información**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere editar la información. El usuario podrá modificar el texto ingresado.

Cuadro 3.31: Caso de uso - Editar Información

Nombre	**Listar cuentas de usuario**
Actor	Usuario administrador
Descripción	Comienza cuando el usuario quiere conocer las cuentas de usuario creadas. Se proporciona entonces una lista de las cuentas de usuario, mostrando correo electrónico, y si la mismas han sido confirmadas o no.

Cuadro 3.32: Caso de uso - Listar cuentas de usuario

Nombre	**Editar cuentas de usuario**
Actor	Usuario administrador
Descripción	Comienza cuando existe alguna dificultad para confirmar la cuenta, esto puede ocurrir si por alguna razón el mail de confirmación no le llega al usuario, entonces se quiere ante estos casos poder confirmar cuentas de usuario desde el administrador.

Cuadro 3.33: Caso de uso - Editar cuentas de usuario

3.3. Página web

Dada la necesidad de concentrar en un sitio información sobre la aplicación, el proyecto, el equipo que formó parte de este proyecto, así como también mencionar aspectos técnicos del trabajo realizado, fue que se pensó en desarrollar una página web, el desarrollo de la misma era deseado pero de baja prioridad. También era deseado que la misma sea *responsive*, es decir, que sea legible en cualquier dispositivo mediante el cual se accediera (computadoras, tabletas, celulares con distintos tamaño de pantalla). A continuación dejamos descritas en casos de uso todas aquellas funcionalidades que fueron deseados contener en la página web.

Nombre	**Descripción de la aplicación**
Actor	Usuario
Descripción	Comienza cuando el usuario quiere obtener información sobre la aplicación. Se quiere que en la pantalla principal, se describa la aplicación mostrando imágenes de la misma, y mencionando todos las funcionalidades que posee.

Cuadro 3.34: Caso de uso - Descripción de la aplicación

Nombre	**Acceso a la descarga de aplicación**
Actor	Usuario
Descripción	Comienza cuando el usuario quiere acceder a la descarga de la aplicación. Se dispone de dos enlaces, uno para la descarga en dispositivos Android y otro para dispositivos iPhone, los mismos redirigirán al usuario al Play Store o App Store, respectivamente.

Cuadro 3.35: Caso de uso - Descarga de la aplicación

Nombre	**Descripción del equipo**
Actor	Usuario
Descripción	Comienza cuando el usuario quiere obtener información sobre la conformación del equipo. Para esto el usuario accede a la opción del menú *Equipo*, la cual lo redirigirá a una nueva pantalla, en donde se detalla las raíces y razones de la existencia del proyecto, así como también los integrantes y las áreas que lo conformaron.

Cuadro 3.36: Caso de uso - Equipo

Nombre	**Descripción del proyecto**
Actor	Usuario
Descripción	Comienza cuando el usuario quiere obtener información sobre el proyecto. Para esto el usuario accede a la opción del menú *Proyecto*, lo cual lo redirigirá a una nueva pantalla, en donde se detalla aspectos técnicos del proyecto, así como también se brinda un acceso al repositorio donde se encuentra alojado el código de la aplicación.

Cuadro 3.37: Caso de uso - Proyecto

Nombre	**Contacto**
Actor	Usuario
Descripción	Comienza cuando el usuario quiere contactarse con el equipo. Para esto el usuario accede a la opción del menú *Contáctenos*, lo cual lo redirigirá a una aplicación de correo que use como predeterminada en su dispositivo, y podrá realizar la consulta que desea mediante el envío de un mail.

Cuadro 3.38: Caso de uso - Contacto

Nombre	**Prensa**
Actor	Usuario
Descripción	Comienza cuando el usuario quiere conocer aquellas publicaciones de medios de comunicación donde se ha mencionado el proyecto, por lo que se le listará todas las imágenes, vídeos y/o accesos directos a las publicaciones.

Cuadro 3.39: Caso de uso - Prensa

3.4. Conclusión

De las funcionalidad descriptas anteriormente solo los casos de uso, *Datos de medicamento* [3.9], *Agregar Medicamento* [3.11], *Mostrar mapa* [3.15], *Descargar Reporte* [3.29], *Prensa* [3.39], no fueron llevados a cabo. Es decir que de los casos de usos propuestos para la aplicación fueron realizados 13 de 16 (81 %), mientras que para el administrador fueron finalizados 18 de 19 (95 %), y para la página web fueron realizados 5 de 6 (83 %). Los casos de uso no realizados eran o bien de baja prioridad o no se tenía información necesaria para realizarlo, un ejemplo de esto son los casos de uso *Datos de medicamento*, *Agregar Medicamento*, *Mostrar mapa*, los cuales eran casos de uso que desde un principio tuvieron alta incertidumbre, y no se llegó a obtener la información necesaria para desarrollarlos. En promedio fueron realizados 88 % de los casos de uso, lo cual consideramos un resultado exitoso, ya que fueron implementados todos los casos de uso de alta prioridad y además desde un principio sabíamos que algunos de ellos no iban a poder ser implementados, por lo que cubrimos la mayor cantidad de casos de usos posibles para el tiempo de desarrollo dado.

En algunos casos se cataloga al prototipado con bosquejos como una práctica costosa, ya que los mismos no podrán ser usados como base de implementación, sin embargo, podemos decir que pese a que la construcción de los bosquejos requirió tiempo, fueron de gran ayuda, nos permitieron entender cómo sería a grandes rasgos la apariencia y la percepción de la interfaz de usuario. A su vez, el equipo pudo entender bien que se quería y de qué iba a tratar el producto cuando se plasmaron todas las ideas en ellos. La creación de prototipos en conjunto con la descripción de los casos de uso, no solo nos ayudó a plasmar las ideas que se tenían por parte de todos los integrantes del equipo y llevarlo a una idea en común, sino que también nos ayudó a entender la arquitectura del sistema, los riesgos tecnológicos, y el tiempo de desarrollo necesario para realizar el producto. Se podría decir entonces, que el tiempo consumido para la creación de los bosquejos, fue más tiempo invertido que gastado, se obtuvo una idea para el diseño inicial, sin haber dedicado tiempo en implementarlo, los cambios en los prototipos no significaron un impacto mayor, cambios que sí hubiesen impactado negativamente si se hubiesen hecho una vez empezado el desarrollo de la aplicación.

Por otro lado, dado que los casos de usos tienen un formato simple y estructurado, fue sencillo para que todos los integrantes de equipo, inclusive aquellos que no eran desarrolladores, entendieran los mismos. Los casos de uso nos permitieron describir fácilmente la interacción del sistema con los demás actores, así como qué datos íbamos a necesitar.

Concluimos entonces, que tanto los prototipos, como la descripción de los distintos escenarios detallados en los casos de uso, nos permitieron acotar los riesgos lo máximo posible, sobre todo aquellos que estaban relacionados a la interfaz y funcionamiento de la aplicación.

Capítulo 4

Diseño

En este capítulo se describen la arquitectura y el modelo relacional propuestos para los requisitos planteados en el Capítulo 3.

4.1. Arquitectura

La arquitectura del sistema construido se divide en cuatro módulos, estos son, la aplicación móvil, el servidor, la base de datos y el sitio web. Esto módulos son independientes, por lo que cualquiera de ellos podría intercambiarse y no afectaría el funcionamiento de los demás. Estos módulos se muestran en la Figura 4.1.

El backend fue desarrollado utilizando Django, un framework de Python, exponiendo sus servicios a través de una API, para la creación de la misma, se hizo uso de Django Rest Framework que serializa los datos del ORM de Django permitiendo el acceso y las actualizaciones.
Por otro lado, también se conecta a la base de datos haciendo uso de Django ORM, la cual utiliza el framework de Django por defecto. Django ORM crea y administra los módulos y las queries de base de datos. El backend también cuenta con un administrador, para la creación del mismo, Django usa los metadatos de los modelos para crear automáticamente la interfaz, que permite crear, ver, actualizar y eliminar registros.

La base de datos, es una base de datos Postgres SQL, que tiene una extensión llamada POSTGIS. En la misma se alojarán todos los datos de los usuarios de la aplicación, así como también todos los datos palinológicos, es decir, aloja los tipos polínicos y fúngicos, las agrupaciones de estos tipos, los reportes de polen y esporas de hongos realizados diariamente por el equipo de palinología.

La aplicación móvil fue desarrollada con el framework de JavaScript llamado React Native, la misma se encuentra disponible tanto para iOS como para Android. La aplicación móvil puede consumir (GET) así como también actualizar (POST, PUT, DELETE) los datos alojados en la base de datos a través de la API REST expuesta por el backend, que hace de intermediario entre la aplicación y la base de datos, actuando entre otras cosas como capa de control.

El sitio web fue desarrollado utilizando Angular, también framework de JavaScript, el mismo actualmente es un sitio estático, pero podría a futuro consumir datos del backend.
Estas tecnologías y otras herramientas serán profundizadas en el Capítulo 6.

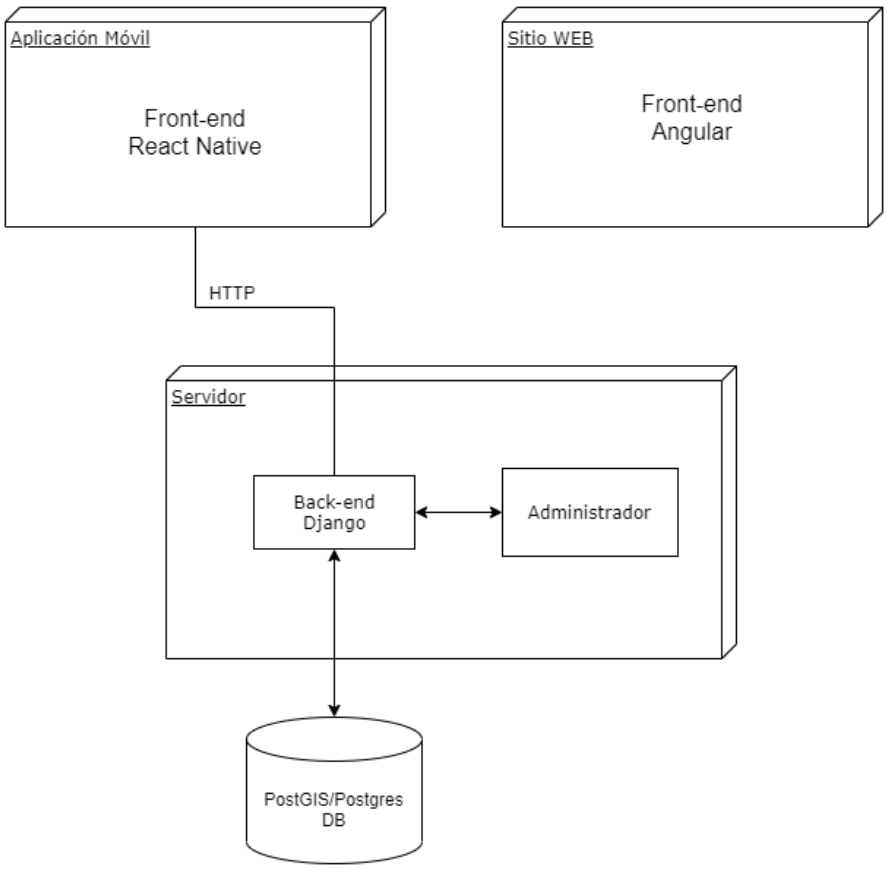


Figura 4.1: Arquitectura

4.2. Modelo relacional

En la figura 4.2 se presenta el modelo relacional utilizado, aquí podemos ver cómo fueron modeladas cada una de las entidades. A continuación se detalla cada una de las entidades, describiendo sus atributos más importantes.

- **User**, esta tabla modela a los usuarios del sistema, incluyendo datos básicos de los mismos, como *nombre*, *email*, *contraseña*, *fecha de nacimiento*, *sexo*, *barrio* y *departamento*. Además de los datos generales, se cuentan con otros atributos, uno de ellos es *is_ superuser* el cual es un atributo booleano, que en caso de tener valor igual *true* significa que dicho usuario tiene acceso de administrador, también cuenta con el atributo *necesita_ onboarding*, atributo también booleano, que por defecto tiene valor *true* y tomará al valor *false* solo si el usuario responde el cuestionario que se le muestra la primera vez que inicia

sesión en la aplicación. Por último, el atributo *onesignal_player_id* permite identificar el dispositivo con el que el usuario inició sesión, con el fin de luego poder enviar notificaciones al mismo.

- **Email Confirmation, Email address y Token**, son tablas creadas por la librería dj-rest-auth de Python, usada para confirmar los emails de los usuarios. El atributo *verified* de la tabla *emailaddres* es un atributo booleano que se establece por defecto en *false*, una vez que el usuario se registra se le envía un mail, el cual contiene un enlace, y si el usuario lo selecciona , entonces el atributo *verified* tomará valor *true*, solo en este estado, el usuario podrá iniciar sesión a la aplicación. Para dar por verificado el email al seleccionar el enlace, se corrobora el token que hay en él, buscando en la tabla *token*, y de aquí se establece a qué usuario pertenece.

- **Notificación**, esta tabla modela las notificaciones a la que los usuarios se pueden suscribir, conteniendo los atributos *id* y *texto*. Por otro lado, la tabla ***user_notificaciones*** es una tabla que modela la relación N a N, entre la tabla *User* y *Notificación*, pudiendo los usuario tener entre 0 y N notificaciones asociadas a su cuenta.

- **Categoría Síntoma**, esta tabla aloja las distintas categorías a las que puede pertenecer un síntoma, la misma tiene los atributos *identificador* y *pregunta*, el atributo *pregunta* es el texto que se le muestra al usuario para saber si ha tenido algún tipo síntoma referente a esta categoría, las mismas se muestran al usuario la primera vez que inicia sesión, el atributo *identificador* es utilizado para mostrar junto con la pregunta imágenes o iconos relacionados a la misma.

- **Síntoma**, esta tabla modela los distintos síntomas que pudo haber tenido el paciente en el último año. Los síntomas perteneciente a una categoría son mostrados si el usuario responde afirmativamente a la pregunta de la misma.

- **Historial**, esta tabla representa el historial del paciente, un historial puede tener varios síntomas, para representar esto, es que existe la tabla *historial_sintomas* que representa la relación N a N entre Historial y Síntoma.

- **Pregunta**, en esta tabla se guardan las distintas preguntas que deberá responder el usuario a la hora de crear un evento en el diario de síntomas, por tanto esta tabla tiene el atributo *pregunta*, que representa el texto que se le mostrará al usuario, y tiene además, una serie de posibles respuestas, un mismo tipo de respuesta puede ser utilizado por varias preguntas, es por esto que se modela con una tabla aparte llamada **tiporespuesta**.

- **Entrada diario**, esta tabla alojará los distintos eventos ingresados por parte del paciente en su diario de síntomas, la misma contiene los atributos *fecha, coordenadas, comentario*, el atributo *fecha* está establecido por defecto en la fecha actual pero puede modificarse. A su vez, al ingresarse un evento en el diario, se deberán guardar los distintos síntomas que tuvo el paciente, esto se guarda en la tabla *valorrespuesta* que modela la relación N a N, entre *entradadiario* y *pregunta*, es decir que un evento en el diario tendrá una respuesta

para cada una de las preguntas que se hagan en el formulario, esta tabla además de tener las claves foráneas a las tablas que relaciona *entrada_diario_id y pregunta_id*, tiene el atributo *respuesta* que guarda la respuesta dada por el paciente a la pregunta en cuestión.

- **Grupo Polínico**, esta tabla modela las categorías a las que un tipo polínico o fúngico puede pertenecer, contiene los atributos *nombre, unidad_medida, nivel_alto, nivel_medio, nivel_alto_sumatoria, y nivel_medio_sumatoria*, atributos necesarios para el procesamiento de los reportes de concentración de polen y esporas de hongos en el aire.

- **Polen**, esta tabla modela los distintos tipos polínicos y fúngicos que se ha comprobado que afectan a los pacientes con rinitis. Esta tabla cuenta con los atributos *tipo, nombre_cientifico, nombre_comun, familia, pricks_presente, alergenicidad, nivel_alto, nivel_medio, nombre_alergia, archivo_pdf*, también tiene el atributo *grupo_polínico_id* que es una clave foránea a Grupo polínico, ya que cada tipo polínico o fúngico, pertenece a un Grupo polínico.

- **User Alergias**, la tabla *user_alergias*, es la tabla que modela la relación N a N entre la tabla **User** y la tabla **Polen**, esta tabla representa el hecho de que un usuario puede ser alérgico a distintos tipos de polen o esporas de hongo. Con esta información es posible dar alertas personalizadas, indicando que se tomen precauciones por parte del paciente cuando existen niveles de concentración de polen o esporas de hongo altas o medias.

- **Reporte Concentración**, esta tabla representa los distintos reportes obtenidos por el equipo de palinología, estos reportes son los obtenidos luego de realizado el muestreo para obtener el nivel de polen y esporas de hongos en aire, estudio que se hace a diario. Esta tabla cuenta con el atributo *fecha*, que tiene por defecto la fecha actual pero podría modificarse, el mismo indica la fecha en que se realizó el reporte; el atributo *tiempo_total*, el cual indica la cantidad de días que se tardó en realizar el muestreo; el atributo *tiempo_extra*, que indica el tiempo extra que se tardó en realizar el muestreo, el mismo podrá ser negativo o positivo; el atributo *porcentaje_muestreado*, este atributo indica el porcentaje del estudio que se tomó en cuenta para obtener el reporte; los atributos *tiempo_real, vol_polen y vol_esporas_hongos* son atributos auto-calculados a través de los demás.
  Con cada reporte se obtiene información sobre cuánto polen o esporas de hongos hay en distintos tipos polínicos y fúngicos, esto se representa mediante una relación N a N entre las tablas ***reporteconcentracion*** y ***polen***, esta relación es implementada haciendo uso de la tabla ***nivelespolen*** que además de tener una clave foránea a cada uno de las entidades que relaciona, (*polen_id* y *reporte_id*), tiene el atributo *nivel* que representa el nivel de polen o espora de hongo hallado para ese tipo polínico o fúngico para un reporte en específico, el atributo *nivel_calculado*, es un atributo auto calculado que se genera a partir del atributo nivel y otros atributos del reporte. A continuación se muestra los

cálculos para hallar los atributos auto-calculados.

$$tiempo\_real = (tiempo\_total + tiempo\_extra) * porcentaje\_muestreado \tag{4.1}$$

$$vol\_polen = (2{,}2 * 2400 * tiempo\_real * \pi * 8{,}21 * 0{,}159)/1000000 \tag{4.2}$$

$$vol\_esporas\_hongos = (2{,}2 * 2400 * tiempo\_real * \pi * 8{,}21 * 0{,}159)/6000000 \tag{4.3}$$

Por último, dependiendo de si se trata de un tipo polínico (ecuación 4.4) o fúngico (ecuación 4.5), el atributo *nivel_calculado* se calcula de la siguiente forma:

$$nivel\_calculado = nivel/vol\_polen \tag{4.4}$$

$$nivel\_calculado = nivel/vol\_esporas\_hongos \tag{4.5}$$

Todas estas ecuaciones fueron dadas por el equipo de palinología, antiguamente todos estos cálculos eran realizados en un Excel, por lo que con estas funcionalidades se reduce el trabajo de este equipo. El nivel calculado es el nivel que se usa para determinar si el tipo polínico y fúngico tiene nivel alto, medio o bajo (nivel que ven los usuarios en la aplicación), dependiendo de las franjas establecidas para cada uno de ellos, es decir que se determinará que un tipo polínico o fúngico tiene nivel alto, si su nivel calculado es mayor al atributo *nivel_alto*, diremos que tiene nivel medio si es mayor al atributo *nivel_medio*, en otro caso diremos que tiene nivel bajo.

- **Términos y condiciones**, en esta tabla se guardan los textos referentes a los términos y condiciones, con el fin de que puedan ser editados por cualquier integrante del equipo y expuestos en los distintos sitios web.

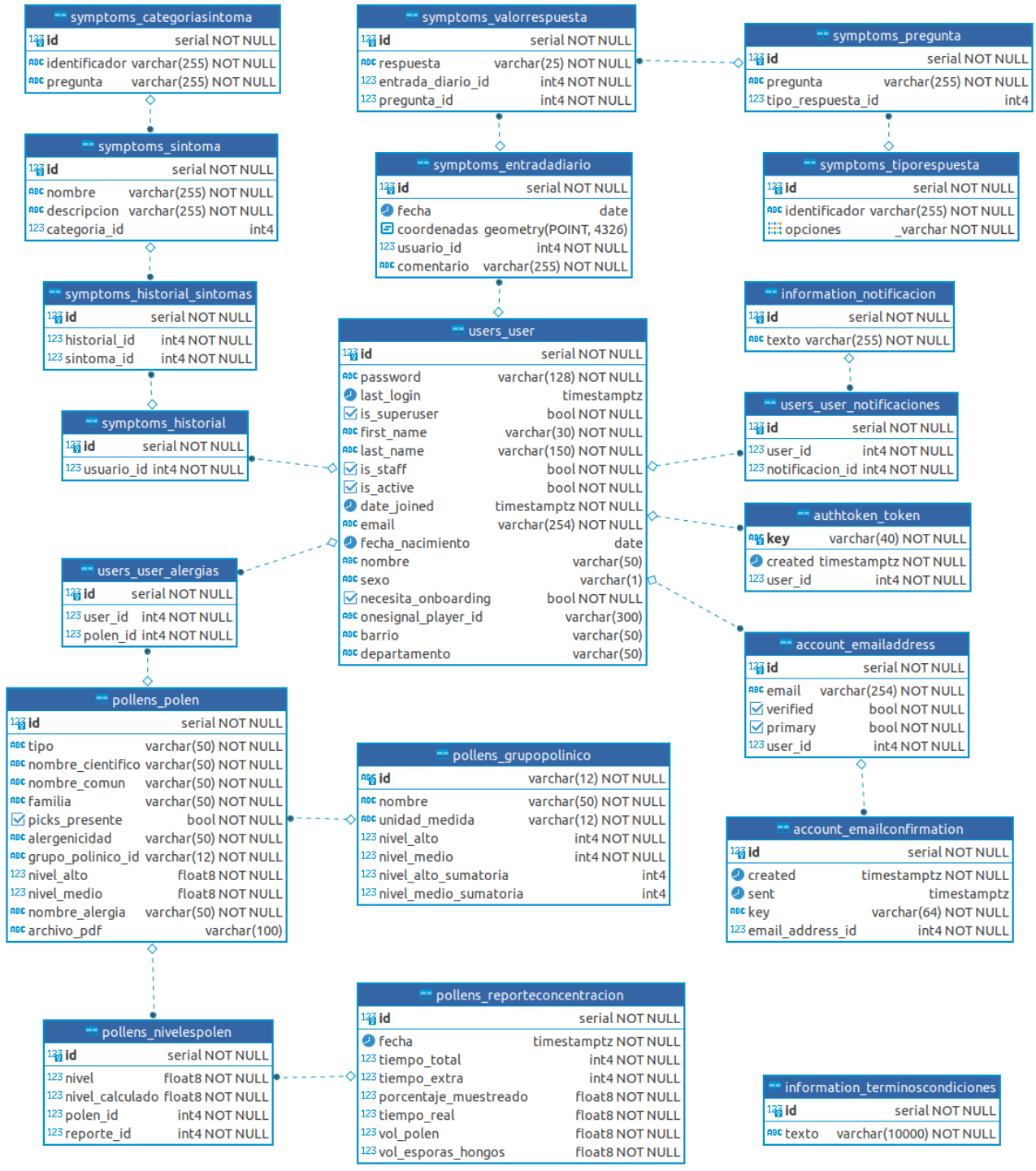


Figura 4.2: Modelo Relacional

4.3. Conclusión

Como podemos ver el sistema cuenta con una arquitectura simple, con bajo acoplamiento entre sus módulos, por lo que sería fácil integrarse con otros módulos o

sustituir cualquiera de ellos. En el Capítulo 3 se mencionaron todas las funcionalidades deseadas, sin embargo los casos de uso *Datos de medicamento* [3.9], *Agregar Medicamento* [3.11], *Mostrar mapa* [3.15], no fueron realizados e implementarlos en un futuro implicaría cambios en la arquitectura. Para el caso de las funcionalidades en torno a datos de medicamentos lo que respecta a arquitectura sería bastante sencillo, agregando una tabla que aloje los medicamentos usados para los tratamientos de rinitis alérgica y otra tabla para representar la relación N a N entre los pacientes y los medicamentos, estaría completa la arquitectura, lo que puede dificultar estas funcionalidades, es cómo cargar la tabla de medicamentos, en este caso sería necesario contar con información certera sobre las drogas y con qué nombres se conocen en el mercado, dado que distintas farmacéuticas fabrican la misma droga con distinto nombre o marca. Para el caso de la funcionalidad "mapa" hoy en día nuestra base de datos de PostgresSQL cuenta con la extensión PostGIS, que da la posibilidad de trabajar y manipular fácilmente datos geográficos, por tanto sería posible agregar en una tabla con datos del arbolado de Montevideo, guardando esta información como polígonos donde los puntos de los mismos sean coordenadas, el hecho de poder guardar coordenada hacen que desplegar estos datos en un mapa sea sencillo. La dificultad de trabajar con datos del arbolado de Montevideo está dada por la obtención de los mismos, el hecho de que no todos los árboles producen alergias, por lo que habría que filtrarlos y además habría que llevarlos a un formato apto para su almacenamiento y posterior lectura.

Capítulo 5

Implementación

En este capítulo se detallan las decisiones de implementación para cumplir con los requerimientos y el diseño.

5.1. Aplicación MIA

MIA, del acrónimo Mi Alergia, es una aplicación pensada para asistir al usuario alérgico. Como se mencionó anteriormente, esta aplicación fue pensada y creada por profesionales de distintas áreas, entre ellas el área de la medicina, la biología y la ingeniería.

5.1.1. Autenticación

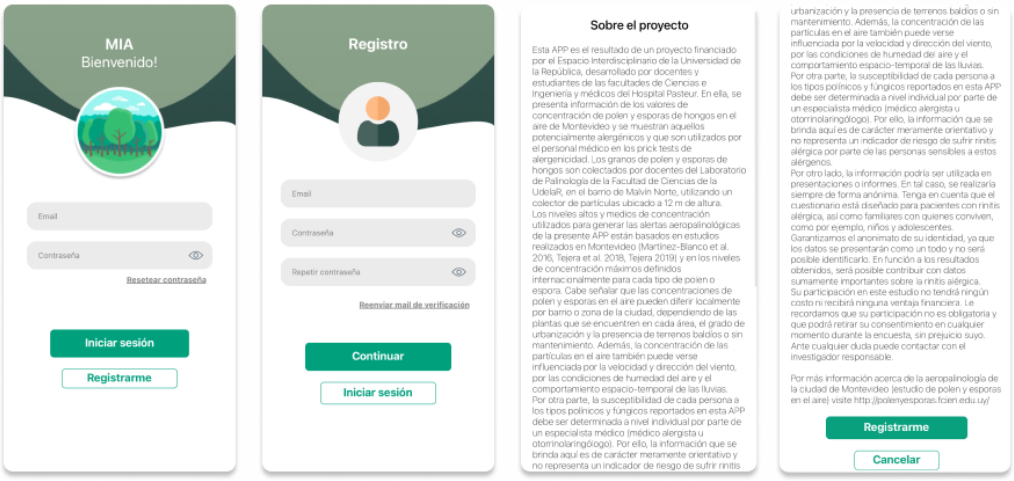
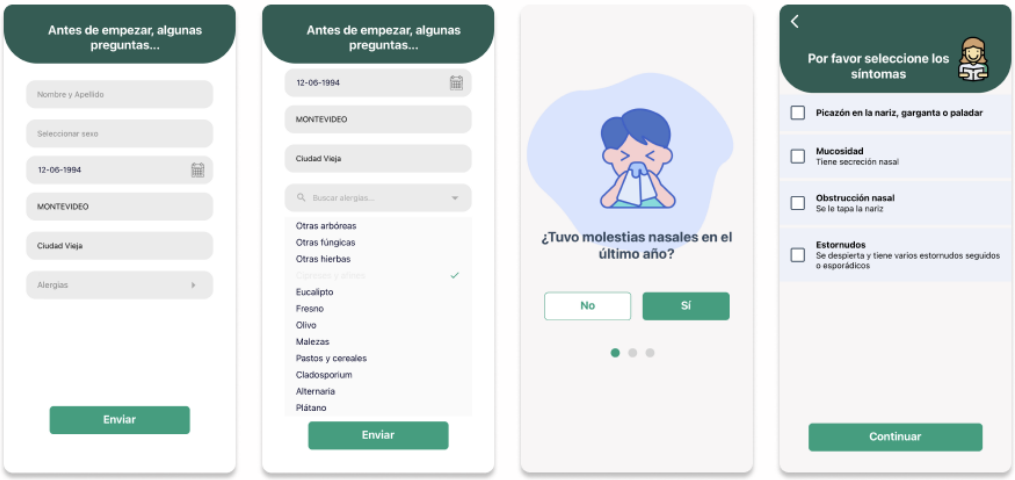


Figura 5.1: Aplicación MIA

Para hacer uso de la aplicación los usuarios deberán registrarse, proporcionando un mail, una contraseña, e indicando que aceptan los términos y condiciones deta-

llados. Estos campos son requeridos, es decir que tanto el frontend, como el backend controlan que se hayan ingresado para poder llevar a cabo el registro. Luego de haber agregado al usuario en la base de datos, se le envía un mail para que termine de completar su registro. El mail contiene un enlace, y accediendo a él se activará la cuenta, esto es posible ya que dicho enlace contiene un token auto-generado por la librería dj-rest-auth de Python, esta librería no solo se encarga de generar el token, sino también de chequear para qué usuario fue generado y validarlo. En caso de que el mail no le llegue al usuario, este podrá reenviar el mail de confirmación nuevamente.
Una vez que los usuarios hayan terminado todos los pasos del registro, estarán habilitados a iniciar sesión con sus credenciales (mail y contraseña), en cualquier dispositivo, conservando todos los datos ingresados. Por otro lado, en caso de que el usuario olvide su contraseña, podrá restablecer la misma ingresando su mail, con lo cual el sistema le enviará un correo electrónico con un enlace que lo redirigirá a una página donde el usuario podrá restablecer una nueva contraseña.

5.1.2. Primer inicio de sesión



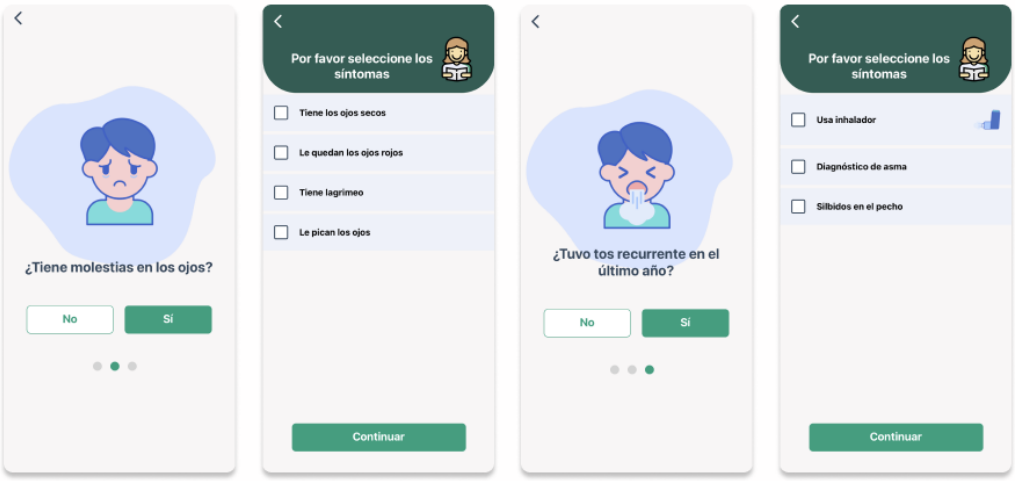


Figura 5.2: Aplicación MIA - Inicio de sesión

La primera vez que el usuario inicie sesión se le pedirá que complete su perfil, además de contestar una serie de preguntas de interés para conocer cómo se ha sentido en el último año.

La primer pantalla que deberá completar será la de datos personales, en la que se le pedirá nombre, sexo, fecha de nacimiento, departamento donde vive, y en caso que el departamento que selecciona sea Montevideo, se le pedirá que ingrese su barrio.

Además, el usuario podrá seleccionar las alergias que posee, en caso de conocerlas, es importante para esta selección haberse hecho el test de alergias (Prick) con un médico, pero dado que no todos los usuarios lo han realizado, este campo no es obligatorio, es decir que se puede continuar sin haber seleccionado ninguna alergia, cabe destacar que la experiencia será mejor si se conocieran estos datos, ya que se mostrarán alertas personalizadas a las alergias que el usuario padezca.

Los datos personales son datos de interés para las médicas, es por esto que los campos nombre, fecha de nacimiento, departamento y barrio (en caso de ser Montevideo el departamento seleccionado), son requeridos y controlados tanto por el frontend como por el backend.

Los datos ingresados en esta pantalla podrán ser actualizados accediendo a las pestaña de ajustes y seleccionando la opción *Información personal*.

En tanto a las preguntas de cómo se ha sentido en el último año serán:

- ¿Tuvo molestias nasales en el último año?
- ¿Tuvo molestias en los ojos en el último año?
- ¿Tuvo tos recurrente en el último año?

A lo que el usuario podrá responder sí o no, y en caso de que conteste que afirmativamente, se le desplegarán varios síntomas para que pueda seleccionar todos aquellos que haya tenido.

5.1.3. Concentración de polen y esporas de hongos

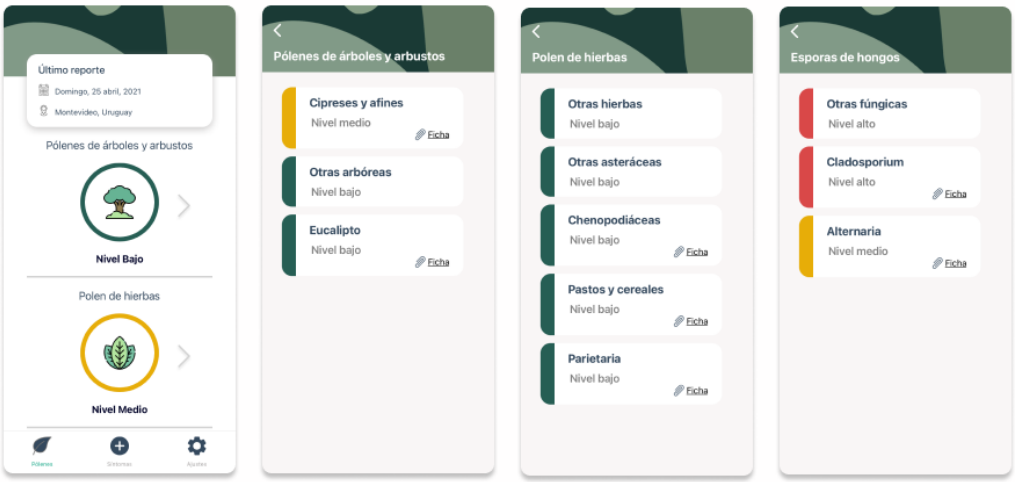


Figura 5.3: Aplicación MIA - Concentración de polen y esporas de hongos

En MIA los usuarios podrán ver los niveles de concentración de polen y esporas de hongos en el aire de Montevideo, los mismos se muestran agrupados por árboles, hierbas y hongos, y podrán ser calificados como alto, medio, o bajo. Esta calificación será visualmente distinguida mediante el uso de los colores del semáforo, donde el rojo se corresponde con el nivel alto, el amarillo con el nivel medio y el verde se corresponde con el nivel bajo. La información sobre los niveles de polen y esporas de hongos es recolectada por el grupo de palinología en el laboratorio de la Facultad de Ciencias de la Udelar, de forma diaria. Una vez que el equipo de palinología obtiene el reporte, procede a ingresarlo en el administrador web creado para este fin, por tanto los reportes quedan persistidos en la base de datos. De todos los reportes almacenados históricamente, los usuarios accederán al último registrado a través de la aplicación, indicado en la pantalla principal la fecha del mismo. Por más información de cómo se obtienen y calculan los niveles de polen y esporas de hongo consulte la sección Subsección 5.2.3.

Los niveles de polen o esporas de hongos de los distintos tipos de alérgenos que componen cada grupo podrán ser revisados en detalle tocando la flecha que se encuentra a la derecha de cada uno.

Los niveles altos y medios de concentración utilizados para generar las alertas aeropalinológicas están basados en estudios realizados en Montevideo, así como también en los niveles de concentración máximos definidos internacionalmente para cada tipo de polen o espora. Cabe señalar que las concentraciones de polen y esporas en el aire pueden diferir localmente por barrio o zona de la ciudad, dependiendo de las plantas que se encuentren en cada área, el grado de urbanización, y la presencia de terrenos baldíos o sin mantenimiento. Además, la concentración de las partículas en el aire también puede verse influida por la velocidad y dirección del viento, por

las condiciones de humedad del aire, y el comportamiento espacio-temporal de las lluvias.

Por lo dicho anteriormente, la información que se brinda en la aplicación es de carácter orientativo y no representa un indicador de riesgo de sufrir rinitis alérgica por parte de las personas sensibles a estos alérgenos, además la susceptibilidad de cada persona a los tipos polínicos y fúngicos reportados en la aplicación debe ser determinada a nivel individual por parte de un especialista médico (alergista u otorrinolaringólogo).

5.1.4. Diario de síntomas

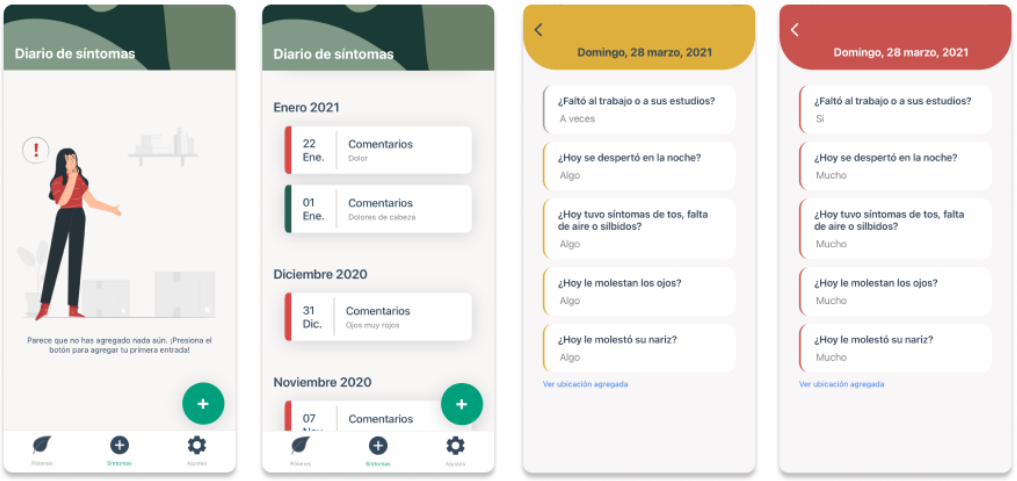
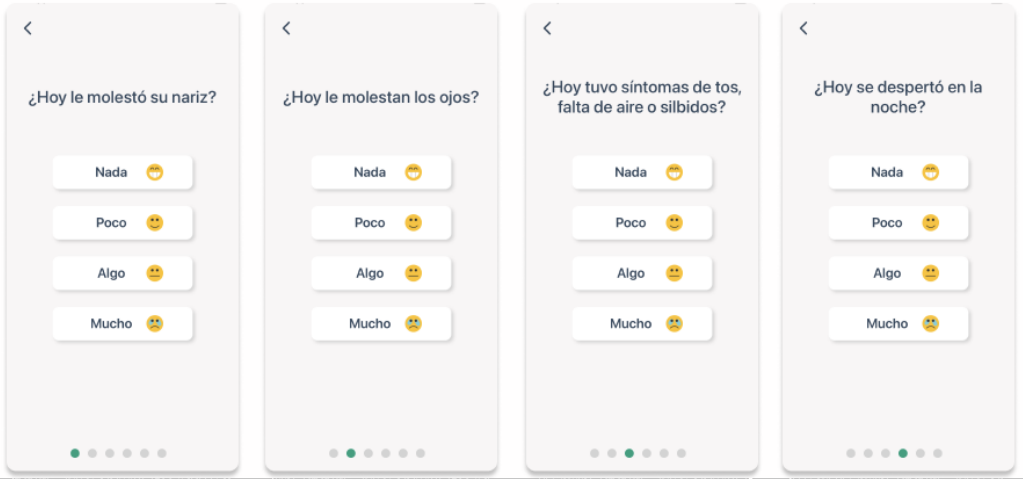


Figura 5.4: Aplicación MIA - Listado de eventos en el diario



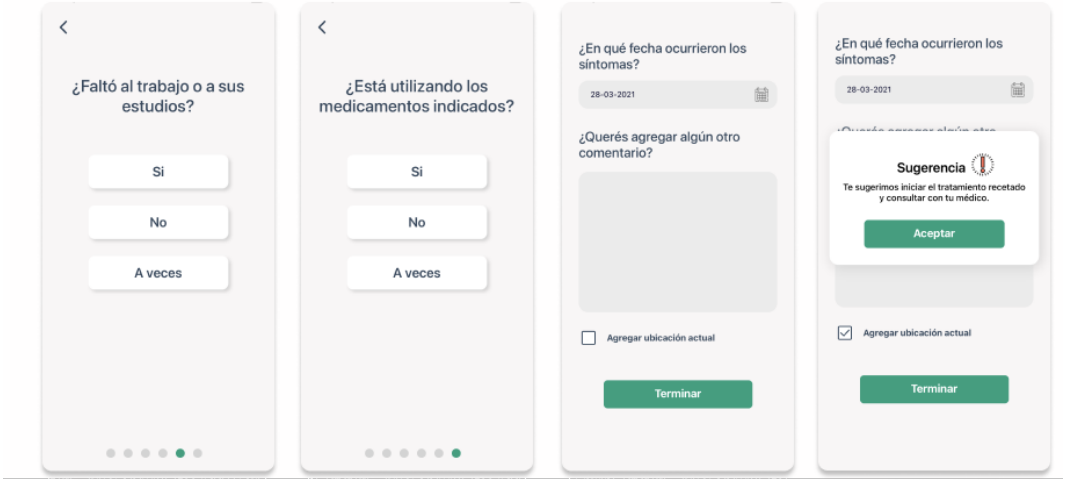


Figura 5.5: Aplicación MIA - Agregado de evento en el diario

MIA cuenta con un amigable diario de síntomas, donde el usuario podrá ingresar cada vez que desee cómo se sintió en el día. El alta de un evento en el diario de síntomas consta de responder una serie de preguntas, pudiendo también agregar un comentario y la ubicación actual donde se encuentra, cabe señalar que para que el sistema acceda a la ubicación del usuario este debe permitirlo, si no desea hacerlo no quedarán guardados datos de ubicación del usuario en el sistema. Las preguntas tienen respuestas predefinidas, y el usuario deberá elegir una de estas. Como se mencionó en la Sección 2.1, las preguntas y respuestas presentes en el diario de síntomas fueron inspiradas en la aplicación Mask Air, las cuales fueron diseñadas por una comunidad de médicos alergistas.

En base a las respuestas dadas por el usuario, los eventos serán clasificados como poco crítico, medianamente crítico y altamente crítico. Todos estos eventos ingresados en el diario serán listados y mostrados al usuario, indicando qué tan crítico fue el mismo, haciendo uso de los colores del semáforo con el fin de que el usuario pueda identificar fácilmente qué días se sintió mejor o peor, pudiendo consultar en detalle qué datos ingresó en cada una de las entradas en el diario.

5.1.5. Alertas personalizadas

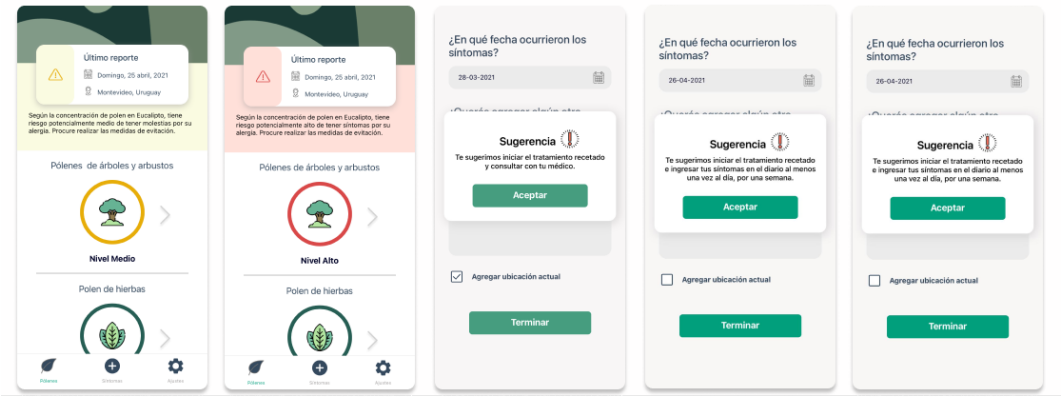


Figura 5.6: Aplicación MIA - Alertas personalizadas

El sistema cuenta con dos tipos de alertas, por un lado las alertas relacionadas a los niveles de polen y esporas de hongo en el aire, y por otro lado las alertas relacionada a la creación de eventos en el diario de síntomas.

Los tipos polínicos y fúngicos potencialmente alergénicos son utilizados por el personal médico en los tests Prick de alergenicidad, por tanto, aquellos pacientes que se hayan realizado estos test, podrán seleccionar en sus datos personales las alergias que padecen. Una vez ingresadas, la aplicación será capaz de alertar de forma personalizada. Estas alertas se dispararán en el caso de que una de las alergias del usuario se vea afectada por los valores de polen o esporas de hongo existentes en el aire.

Por otro lado, cuando el usuario ingrese un nuevo evento en el diario de síntomas, al terminar de responder las preguntas y en caso de que estas indique que el usuario está ante una crisis, se le mostrará una alerta correspondiente. Las alertas tiene tres variantes:

- Poco crítico: cuando contiene una o dos respuestas críticas.
- Medianamente crítico: cuando contiene tres o más respuestas críticas.
- Altamente crítico: cuando se crearon 5 entradas en la misma semana con al menos una respuesta crítica en cada una.

5.1.6. Notificaciones

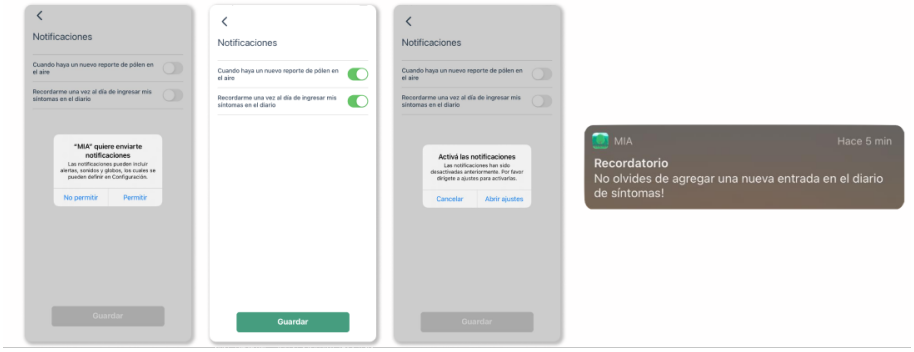


Figura 5.7: Aplicación MIA - Notificaciones

MIA también cuenta con un sistema de notificaciones que el usuario podrá activar si así lo desea, una de ellas es para recibir un recordatorio diario para agregar un nuevo evento al diario de síntomas, la segunda es para recibir una notificación cada vez que se registre un nuevo reporte en el sistema.

Para hacer uso de esta funcionalidad también se requerirá aceptar los permisos de notificación, por lo que las mismas estarán deshabilitadas por defecto hasta que el usuario las active manualmente. Si por alguna razón el usuario había revocado los permisos de notificación anteriormente, al intentar activarlas la aplicación lo ayudará a llegar a la sección de ajustes del dispositivo en donde puede habilitar los permisos (comportamiento análogo a la habilitación de permisos de ubicación).

Si bien hoy en día contamos solamente con dos notificaciones, se creó de forma que los textos mostrados son consumidos de la API, de forma que si en el futuro se quisiera agregar una nueva notificación, solamente se debe implementar la lógica del lado del backend y no requerirá una nueva versión de la aplicación. Este sistema de notificaciones push fue implementado utilizando el servicio OneSignal (por más información ver la Subsección 6.5.3).

5.1.7. Fichas informativas

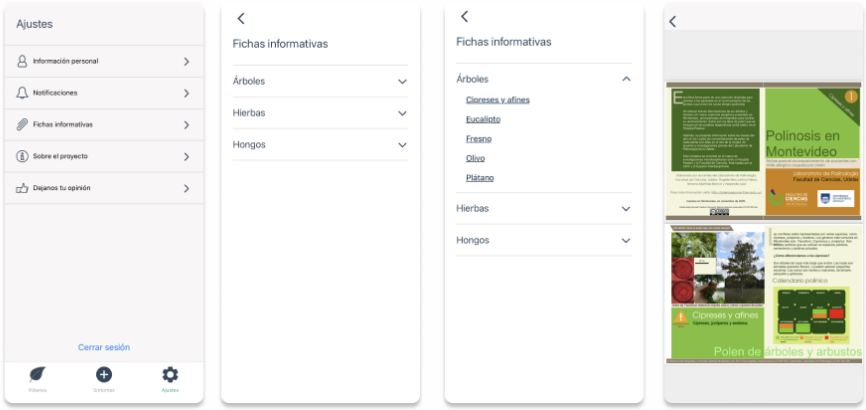


Figura 5.8: Aplicación MIA - Fichas informativas

La aplicación MIA también dispone de fichas informativas sobre los distintos alérgenos, los mismos que se detectan en los test Pricks. Estas fichas fueron diseñadas para orientar a los pacientes en el reconocimiento de las plantas cuyo polen les causa alergia, así como también dar algunas medidas preventivas.

Estas fichas son subidas en el administrador para cada grupo polínico y son mostradas automáticamente por la aplicación, por lo que se pueden borrar, agregar o cambiar sin necesidad de una nueva versión de la aplicación.

En la aplicación se puede acceder a ellas a través del detalle de cada grupo polínico, y también se pueden encontrar agrupadas por tipos polínicos en la pestaña de *Ajustes*, en la sección de *Fichas informativas*.

5.2. Administrador

El administrador es una funcionalidad incluida por defecto en Django, sin necesidad de instalar una librería de terceros. Esta aplicación usa los metadatos de los modelos para crear automáticamente la interfaz, donde se puede crear, ver, actualizar y eliminar registros, con la posibilidad de incluir ciertas customizaciones. En nuestro caso el administrador fue creado para la gestión interna de los datos, sobre todo para la creación de nuevos reportes de medición de polen. Este administrador nos ahorró mucho tiempo de desarrollo, ya que solo necesitamos registrar en un archivo los modelos que queremos mostrar en el administrador. Para poder iniciar sesión en el sitio de administración es necesario primero crear un "superusuario", para esto, se necesita ejecutar el siguiente comando desde el entorno virtual del servidor:

```
python manage.py createsuperuser
```

En nuestro caso creamos superusuarios para los integrantes del grupo de ingeniería y para los integrantes del grupo de palinología. Los modelos que registramos fueron el de *Cuentas*, *Términos y condiciones*, *Pollens*, y *Users*.

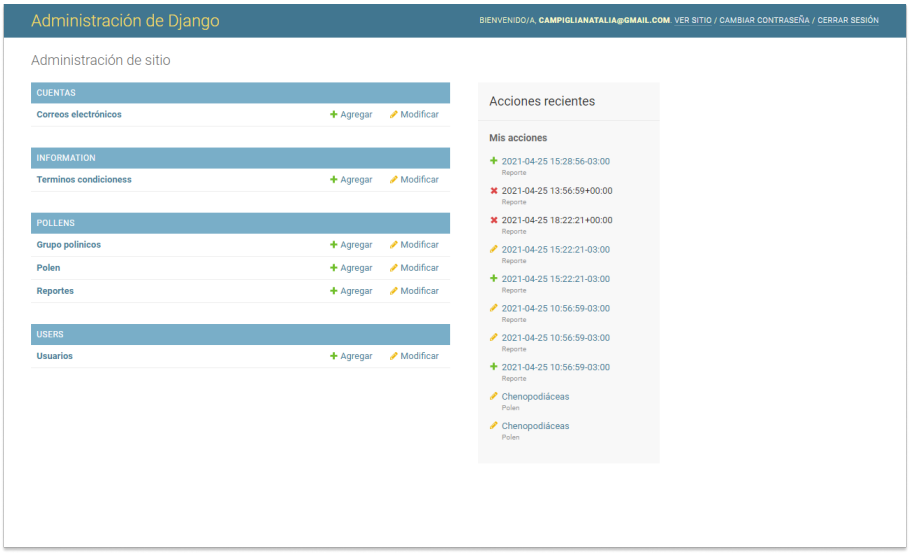


Figura 5.9: Administrados

5.2.1. Cuentas

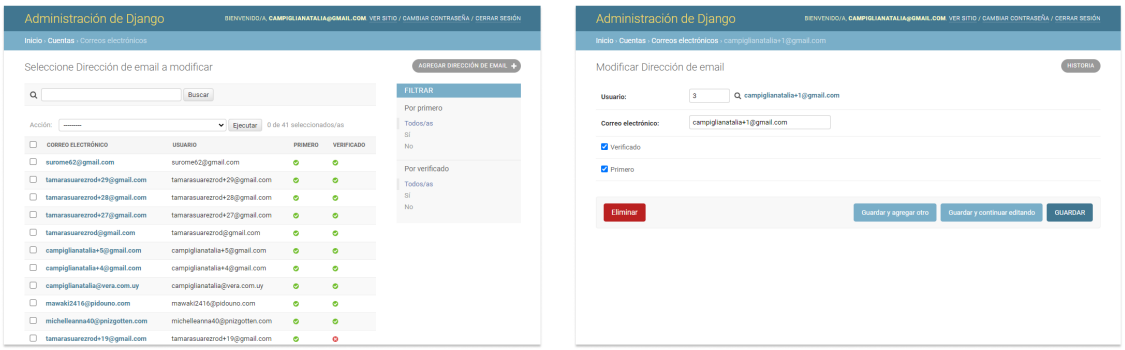


Figura 5.10: Administrados - Información

En la pantalla de *Cuentas* se puede ver todas las cuentas de usuarios creadas y si están confirmadas o no. También se puede crear, editar o eliminar entidades. El editar cuentas nos ha sido de utilidad para usuarios que han tenido problemas con la confirmación de la cuenta, ya que nos permitió confirmarlas rápidamente accediendo directamente desde esta web.

5.2.2. Información

Figura 5.11: Administrados - Cuentas de usuario

Esta pantalla se creó con el objetivo de incluir información del proyecto que se quiera mostrar a los usuarios, ya sea desde la aplicación móvil o desde la página web, con el fin de que esta información sea mostrada y modificada de forma dinámica, es decir, si se quiere actualizar un texto no se necesita actualizar el backend o el frontend, sino que es suficiente con cambiarlo desde el administrador web y los cambios se reflejarán automáticamente, pudiéndolo hacer cualquier usuario con acceso a la página de administración. Actualmente solo se guardan los términos y condiciones de MIA, pero podrían alojarse más textos.

5.2.3. Pólenes

Figura 5.12: Administrados - Polen

En la pantallas *Pólenes*, se encuentran todos los datos referidos a pólenes y esporas de hongos, a su agrupamiento, y a los reportes sobre los mismos.

Grupos polínicos

Los grupos polínicos son los grupos en los que se dividen los distintos tipos polínicos y esporas de hongo.

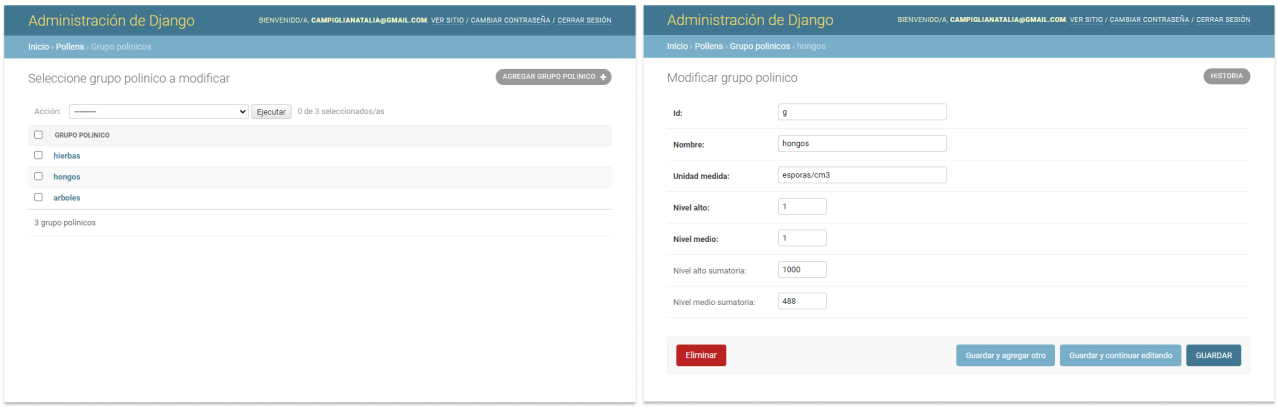


Figura 5.13: Administrados - Grupos polínicos

Actualmente el sistema cuenta con los grupos árboles, hierbas, y hongos, estos grupos son los que ve el usuario en la pantalla principal de la aplicación donde se ve el nivel de polen para cada uno de estos.

Para crear un grupo es necesario llenar los campos *id*, *nombre*, *unidad de medida*, *nivel alto*, *nivel medio*, *nivel alto sumatoria* y *nivel medio sumatoria*, estos campos son todos requeridos, con excepción de *nivel alto sumatoria* y *nivel medio sumatoria*. Los niveles alto y medio, son las franjas que determinarán si el grupo tiene nivel alto, medio o bajo. Entonces, para decir que un grupo tiene nivel alto, medio o bajo, se toma el último reporte, y en base a cuántos de los elementos de su grupo tienen nivel alto, medio o bajo, se determina el nivel del grupo. Por ejemplo, si el nivel alto y medio en el grupo se definen en 1, diremos que el grupo tiene nivel alto si al menos uno de los tipos polínicos y fúngicos que lo componen, tiene nivel alto, en caso de no haber ningún elemento con nivel alto, pero sí al menos uno en nivel medio, entonces el grupo tendrá nivel medio, por último si ninguno de los tipos que componen el grupo tienen nivel medio o alto, entonces el grupo tendrá nivel bajo.

En el caso de que se hayan definidos los valores de *nivel alto sumatoria* y *nivel medio sumatoria*, entonces un grupo podrá tener nivel alto o medio, si la sumatoria de los niveles de cada uno de los tipos polínicos y fúngicos que componen el último reporte y que pertenecen a el grupo en cuestión, supera una de esas franjas.

Tipos polínicos y fúngicos

En esta pantalla se podrán ver todos los tipos polínicos y fúngicos cargados en el sistema, así como también se podrá agregar, modificar o eliminar entidades. Los tipos polínicos y fúngicos agregados son aquellos que el equipo de palinología y medicina consideran que pueden causar alergias o malestar en las personas.

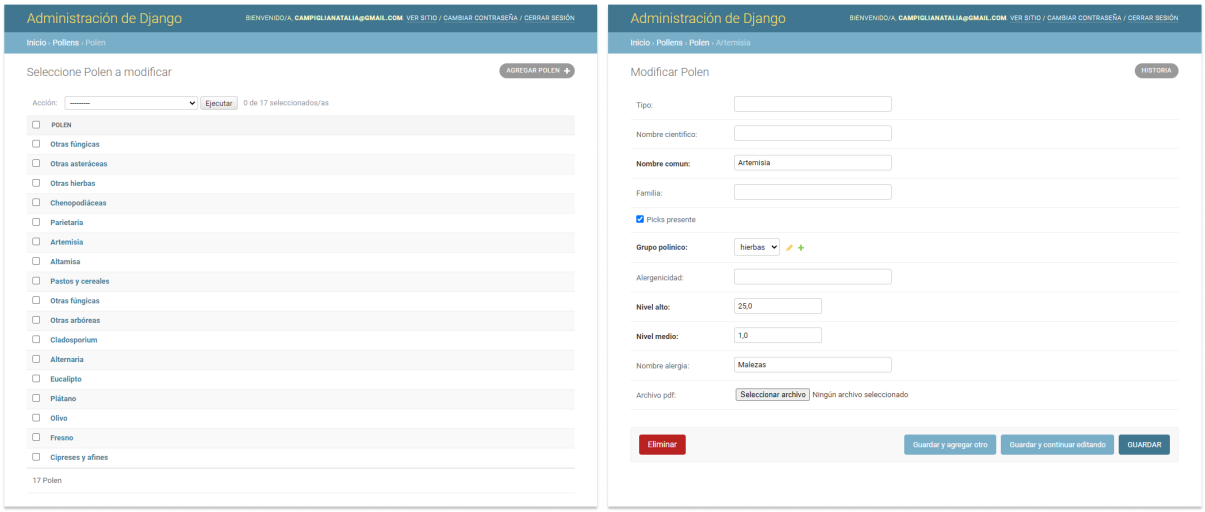


Figura 5.14: Administrados - Tipos Polínicos y Fúngicos

Para crear un tipo polínicos o fúngicos es necesario completar los campos que se ven en el formulario, algunos de ellos no son requeridos ya que no se utilizan actualmente en el sistema, pero a futuro podrían utilizarse. Los campos no son utilizados por el sistema actualmente son “tipo”, “Nombre científico”, “Familia”, y “Alergenicidad”. El campo “Nombre comun” es el nombre con el que se muestra el tipos polínicos y fúngicos en los detalles del grupo en la aplicación, los campos “Niveles alto” y “Nivel Medio”, son las franjas que determinan si el nivel de polen o esporas de hongos recolectadas en el último reporte, es alto medio o bajo, es decir que, si el nivel de polen o espora de hongo calculado en el último reporte para ese tipo polínico o fúngico supera el nivel alto, diremos que este tipo tiene nivel alto, en cambio si no supera el nivel alto, pero sí supera el nivel medio, diremos que este tipo tiene nivel medio, y de no cumplirse ninguna de estas condiciones, diremos que el tipo tiene nivel bajo.

El campo “Picks presente” es una atributo booleano, que indica si el tipo polínico o fúngico se puede detectar realizando los test de Picks (test médico que determina las alergias de un paciente), todos los tipos polínicos y fúngicos que cuenten con “Picks presente” en TRUE son potenciales alérgenos, por lo tanto son estos los que se muestran al usuario en la lista donde puede elegir sus alergias. Muchas veces el nombre de un tipo polínico o fúngico, no es el nombre con el que el usuario conoce su alergia, en caso de que la alergia al tipo polínico o fúngico se le proporcione a los pacientes con un nombre distinto al “Nombre Común” es necesario completar el campo “Nombre alergia”, y este es el nombre con el que el tipo polínico o fúngico aparecerá en la lista de alergias mostradas al usuario en la aplicación. El campo “Grupo polínico” es una clave foránea a uno de los grupos ingresados en el sistema, este campo es obligatorio, y significa que el tipo polínico o fúngico que se está agregando pertenece al grupo indicado en este campo, por lo que actualmente podría pertenecer a árboles, hierbas, u hongos.

Por último, el campo “Archivo PDF”, no es un campo requerido, fue pensado para desplegar en la aplicación un archivo pdf informativo para el grupo polínico específico, de modo que los usuarios puedan tener más información sobre su alergia.

Reportes

En esta pantalla se encuentra el listado de los reportes de polen y esporas de hongo, en la misma también se pueden crear, editar y eliminar nuevos reportes. Estos son cargados diariamente por el equipo de palinología y son consumidos automáticamente por la aplicación.

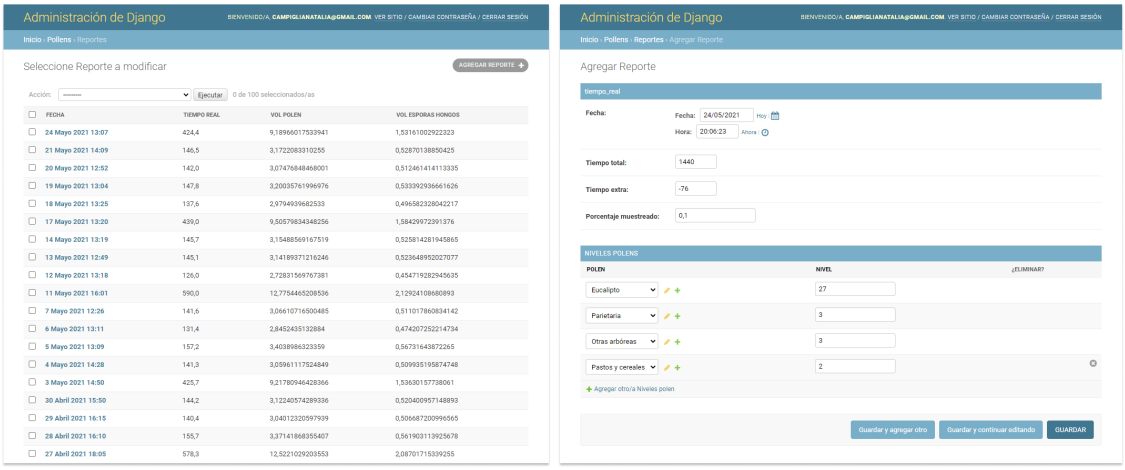


Figura 5.15: Administrados - Reportes

A la hora de crear un reporte nuevo, es necesario completar los siguientes campos del formulario:

- Fecha: es requerido y tiene establecido por defecto la fecha y hora actual pudiéndose cambiar de ser necesario.
- Tiempo total: es requerido y tiene como valor por defecto 1440, correspondiente a la cantidad de minutos que hay en un día, hace referencia a la cantidad de minutos que se demoró en obtener la muestra de polen y esporas de hongo en el aire.
- Tiempo extra: no es requerido y solo se utiliza en caso de que el tiempo para obtener la muestra haya tenido una variación.
- Porcentaje muestreado: es requerido y está establecido por defecto en 0,1, hace referencia a qué porcentaje de la muestra es analizada.

Por otro lado cada muestra realizada por el equipo de palinología involucra distintos tipos polínicos y fúngicos, los que se vean involucrados se agregaran al listado “Niveles de Polen” ingresando en “Agregar otro nivel polen” y buscándolo en la lista

desplegable, para cada uno de los tipos polínicos y fúngicos se deberá ingresar también el nivel de polen y esporas de hongos obtenidos de la muestra.
Todos los valores ingresados serán usados para calcular el Nivel Calculado de cada tipo polínicos y fúngicos ingresado en el reporte, el nivel calculado es distinto del nivel, este último hace referencia a lo registrado por la muestra, mientras el nivel calculado en un valor que se halla mediante los siguientes cálculos:

$$tiempoReal = (tiempoTotal + tiempoExtra) * porcentajeMuestreado \quad (5.1)$$

$$volumenPolen = (2{,}2 * 2400 * tiempoReal * \pi * 8{,}21 * 0{,}159)/1000000 \quad (5.2)$$

$$volumenEsporasHongos = (2{,}2 * 2400 * tiempoReal * \pi * 8{,}21 * 0{,}159)/6000000 \quad (5.3)$$

Por último, dependiendo de si se trata de un tipo polínico (4) o fúngico (5), el nivel calculado se calcula de la siguiente forma:

$$nivelCalculado = nivel/volumenPolen \quad (5.4)$$

$$nivelCalculado = nivel/volumenEsporasHongos \quad (5.5)$$

El nivel calculado es el nivel que se usa para determinar si el tipo polínico y fúngico tienen nivel alto, medio o bajo (nivel que ven los usuarios en la aplicación) dependiendo de las franjas establecidas para cada uno de ellos.

5.3. Página web

La página web de MIA (`https://mialergia.fcien.edu.uy/`) fue creada con el fin de informar sobre la aplicación, el proyecto y el equipo. La misma cuenta con una pantalla de inicio en la que se muestran fotos de la aplicación, y se describe los posibles usos que se le puede dar a la misma.
Luego se cuenta con una sección *Equipo*, en la que se describe cómo surgió el proyecto y quiénes lo integran. También cuenta con una pantalla de *Proyecto*, orientada a un perfil más técnico, en la que se describen brevemente las tecnologías utilizadas, y se agrega además un enlace a la organización de MIA en GitHub, donde se encuentran los tres repositorios de la aplicación móvil, web, y backend, cabe destacar que todo el desarrollo es de código abierto y se encuentra alojado en los mismos.

Por otro lado la aplicación tiene una sección *Contáctenos*, que al seleccionarla se abrirá automáticamente la aplicación de mail utilizada por defecto en el dispositivo desde el cual se esté consultando la página web, teniendo como destinatario prefijado el mail de MIA.

La web contiene un cabezal en el que se ven las distintas secciones a las que el usuario puede acceder, así como también un pie de página con redirección a la descarga de la aplicación MIA tanto para la tienda de Android como para iPhone, esta facilidad es un punto importante de la web dado que muchas personas llegan a la misma a través de la prensa, y desde aquí pueden acceder directamente al sitio donde pueden descargar la aplicación, sin necesidad de buscarla manualmente. También se incluye un enlace a los términos y condiciones del proyecto, y se muestran los logos de todas las entidades participantes del proyecto. Un punto a destacar es que la página web fue diseñada cuidadosamente para que se vea de forma correcta en las distintas pantallas, ya sea desde un celular, tableta o computadora.

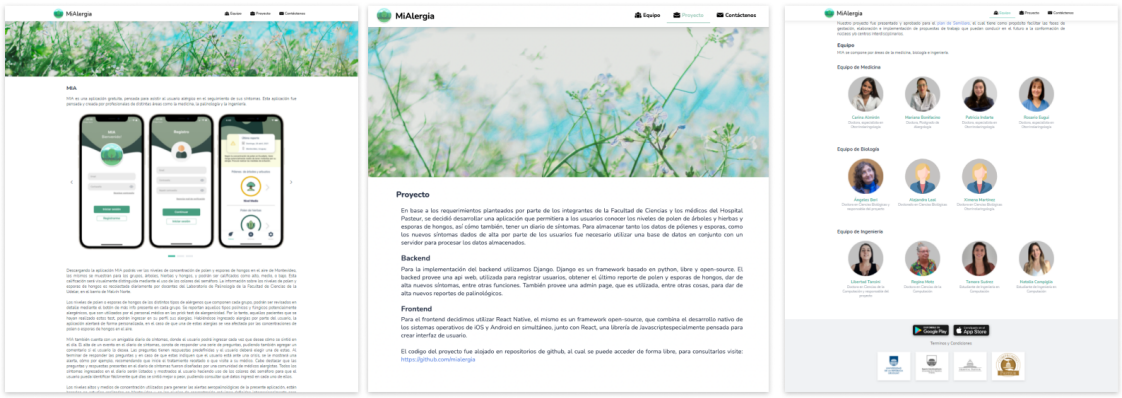


Figura 5.16: Página web de MIA

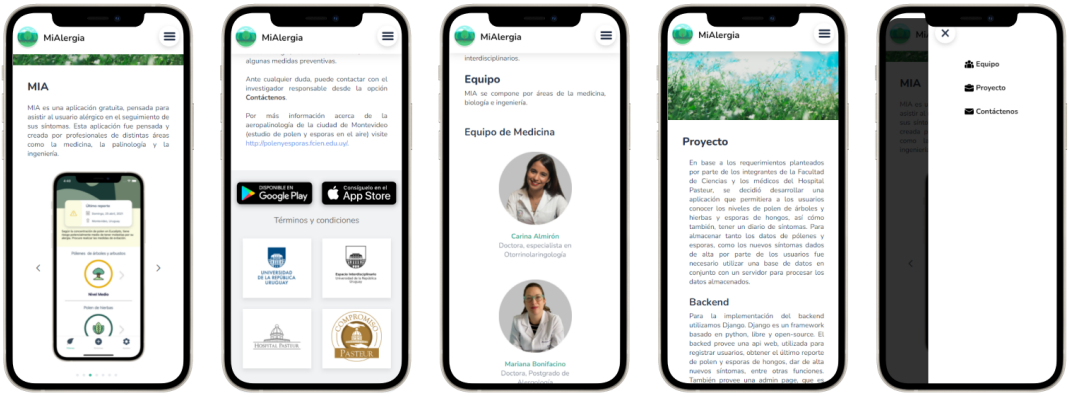


Figura 5.17: Página web de MIA

Capítulo 6

Tecnologías y herramientas utilizadas

En este capitulo se mencionan todas las tecnología y herramientas utilizadas a lo largo del proyecto, con muchas de las cuales no contábamos con conocimiento previo, por lo que tuvimos que capacitarnos previamente a usarlas.

En especifico para el desarrollo de la aplicación se decidió utilizar React Native, esta decisión fue tomada en base a que tiene como ventaja que el código desarrollado puede ser utilizado tanto por el sistema operativos iOS, como por el sistema operativo Android, lo que implico no tener que desarrollar dos aplicaciones en paralelo, sino una sola. Cabe destacar que existen otras librerías que pueden ser utilizadas para el desarrollo de aplicaciones para ambos sistemas operativos en simultaneo, sin embargo React Native es de las mas usadas, por lo que tiene una gran comunidad de apoyo. Por otro lado, uno de los integrantes del equipo ya contaba con experiencia trabajando con la misma, lo cual fue de gran ayuda para el comienzo del desarrollo.

Para el desarrollo del backend se decidió utilizar Django que es un framework del lenguaje Python y aunque el uso de este framwork requirió tiempo de capacitación, ya que ninguno de los integrantes del equipo contaba con conocimiento en esta tecnología, esta decisión fue tomada en base a que el mismo posee una sintaxis simple y fácil de aprender, su arquitectura de MTV implica legibilidad y claridad del código, punto importante pensando en que cabe la posible de que exista una extensión de proyecto y tal vez que otros equipos continuaran el desarrollo. Por otro lado, desde desde el punto de vista de Data Analytics y Big data, al ser Python un modelo de código abierto el mismo admite múltiples plataformas y con diversas librerías que lo convierten en un lenguaje de programación famoso en áreas como la computación científica donde están involucradas operaciones con grandes volúmenes de datos. Estas librerías tienen paquetes que permiten simplificar tareas como: computación numérica, análisis estadístico, visualización gráfica de resultados y métricas, agrupación y cotejación de datos, punto importante para poder desarrollar posible trabajos futuros, como el cruzamiento y análisis de los datos recabados.

Para el desarrollo de la pagina web se decidió utilizar Angular, framework de ???framework, ya que uno de los integrantes del equipo contaba con experiencia en el mismo, además de que permite crear aplicaciones web escalables, por lo que si a futuro se decide agregar funcionalidades en la misma, no tendríamos problemas de

rendimiento. Además angular cuenta con comunidad muy grande lo cual hace que su desarrollo sea confiable.

6.1. Frontend

Para el frontend decidimos utilizar React Native, que combina el desarrollo nativo de los sistemas operativos de iOS y Android en simultáneo, junto con React, una librería de Javascript especialmente pensada para crear interfaz de usuario.

6.1.1. React y React Native

React (`https://reactjs.org/`) es una librería de Javascript, creada y mantenida por Facebook, y al ser de código abierto, es también mantenida por su comunidad, quienes pueden subir mejoras o reportar errores en GitHub (`https://github.com/`). Combina la rapidez de Javascript junto con una nueva forma de renderizar componentes, haciéndolos más dinámicos para la experiencia de usuario con la interfaz.

React es muy bien conocido por utilizar en su implementación el llamado DOM (document object model) virtual, el DOM es una estructura lógica de objetos que tiene forma de árbol, los navegadores web son los encargados de traducir HTML, XHTML, o XML en el DOM. Por lo tanto, cada vez que se modifica un elemento dentro del mismo, deberá ser actualizado junto con todos sus hijos, y estas actualizaciones son costosas, lo que puede influir en el rendimiento de la aplicación. Con el objetivo de optimizarlo, React contiene el concepto de DOM virtual, el cual es una copia en memoria del real, que actúa de intermediario entre la aplicación y el mismo. Así, cada vez que se produce un cambio en el estado de la aplicación, el DOM virtual es actualizado para, a continuación, realizar una comparación entre esta nueva versión y la antigua del DOM real. Una vez detectados los cambios, estos son enviados de golpe al DOM real para realizar una única actualización de la interfaz gráfica. Así es como funciona React y lo que le permite tener este gran rendimiento.

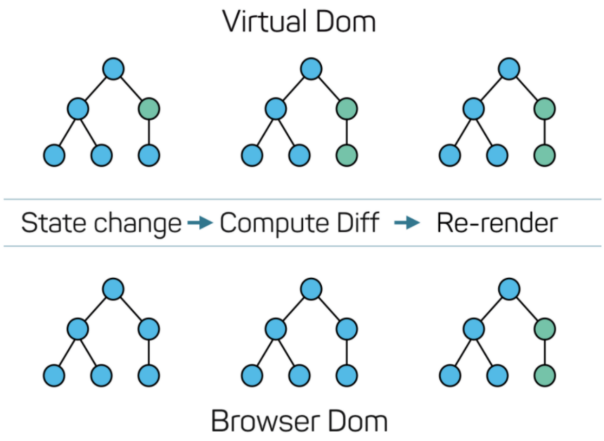


Figura 6.1: React Native

Una de las grandes ventajas de utilizar React Native (`https://reactnative.dev/`), es que al ser cross-platform, hace que un mismo código se renderice automáticamente en los lenguajes nativos de desarrollo iOS (en el lenguaje Objective-c o Swift) y Android (en el lenguaje Java), por lo cual no se necesita tener dos desarrollos en paralelo para poder tener un producto que pueda ser utilizado en ambas plataformas, esto nos dio la posibilidad de llegar a más usuarios, en el límite de tiempo que teníamos. Muchas veces los lenguajes nativos son elegidos cuando se necesita crear una aplicación muy compleja, que requiera personalizar componentes nativos del dispositivo. Sin embargo nuestra aplicación no tenía ninguna de esas necesidades, por lo que React Native era una excelente opción, algunos ejemplos de marcas conocidas que lo utilizan son: Facebook, Walmart, Soundcloud, Bloomerang, y Wix.

6.2. Backend

6.2.1. Django

Django es un framework del lenguaje Python, de código libre, creado específicamente para construir aplicaciones de la manera más rápida posible, siguiendo la idea de que el desarrollador pueda centrarse en crear la aplicación sin tener que reinventar la rueda.

La arquitectura que siguen los productos creados con Django es la arquitectura MTV (Model, Template, View), es una variación del patrón MVC (Model, View, Controller). Los *models* contienen la estructura lógica del proyecto, y son el intermediario y quienes manejan los datos entre la base de datos y las *views*, en el *model* se encuentra la definición de cómo la estructura de datos deberá provenir de las *views* para ser guardada en la base de datos, y vice-versa. Las *views* reciben datos, así como lo hacen los métodos PUT, GET, etc. y formatea los datos a través del *model* para que puedan ser guardados en la base de datos, y así análogo en la otra dirección, se comunica con la base de datos para recibir la información que se la pasará al *template* para que sea mostrada al usuario. El *template* es la parte visible por el cliente, su objetivo es mantener la parte estática de una salida HTML, XML, CSV. En nuestro caso, al tener todo el frontend de la aplicación en su propia tecnología, el uso de templates fue mínimo, solamente hicimos uso de ellos para renderizar el contenido de los mails enviados.

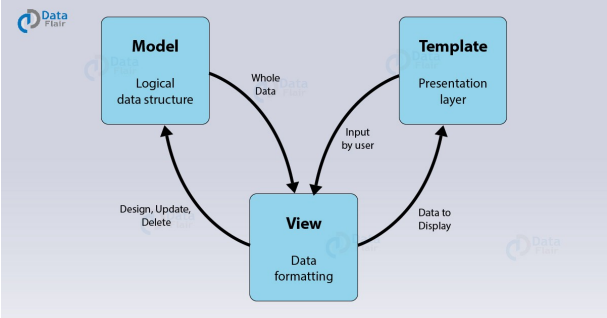


Figura 6.2: Django

6.2.2. Django REST Framework

Una de las librerías principales que usamos en el backend, es django-rest-framework (`https://www.django-rest-framework.org/`) diseñada especialmente para poder crear una API REST (REpresentational State Transfer) con proyectos en Django. Hay muchas formas de crear una API web, básicamente es crear un conjunto de funciones y procedimientos que permiten a una aplicación acceder a los servicios que brinda otra aplicación. Una API REST determina cómo será diseñada, es un conjunto de reglas o guías que se seguirán para crearla, por ejemplo, que se deberá poder acceder a una parte de los datos cuando se acceda a determinada URL, cada una de estas es llamada request, y los datos que son recibidos mediante esta URL, es llamado response. Cada request consta de cuatro componentes: endpoint, método, cabezales, y datos (o body). El endpoint es la URL que mencionábamos anteriormente, el cual mantiene la primer parte siempre estática, y cambia su última parte para determinar a qué datos quiere acceder.

Otra de las grandes ventajas que nos brindan las API REST, es que la interacción entre cliente-servidor comunican sus estados a través de los códigos estándares de HTTP, y esto facilita el trabajo tanto para el desarrollador como para el consumo y manejo de estados del lado del cliente. En conclusión, usar una API de estas características, nos da la posibilidad de acceder a los datos mediante URL's, utilizar el protocolo HTTP, y JSON para el formato de datos. Lo cual resulta en en un desarrollo simple, estándar, compatible con diferentes tecnologías y escalable.

6.2.3. Autenticación con JWT

Para la autenticación de usuarios en la aplicación se utilizó el método JSON Web Token (JWT), un estándar abierto (RFC 7519 [4]) basado en JSON que propone la creación de tokens para identificar a un usuario y sus privilegios, de forma de transmitir de una manera segura la información entre el usuario y el servidor, como un objeto JSON, esta información puede ser verificada porque está digitalmente firmada. Para esto utilizamos la librería Simple JWT creada como plugin para Django REST Framework que provee dicha autenticación.

Funcionamiento

Cuando un usuario se autentica satisfactoriamente con sus credenciales (en nuestro caso, mail y contraseña), se devuelve dos tipos tokens en la respuesta:

- **Access token:** se usará cada vez que se quiera acceder a información protegida. En cada request que requiera permisos de autenticación, se deberá enviar el access token correspondiente al usuario, el cual fue guardado en el frontend cuando el usuario se autenticó satisfactoriamente. Es un token con tiempo de vida corto que puede ser configurado según la necesidad, en nuestro caso le asignamos un tiempo de vida de 10 minutos. Cuando su tiempo de vida se termina, se utiliza el refresh token para obtener uno nuevo, este proceso es transparente para el usuario, ya que este solamente pide información, y el frontend se encarga de obtener un access token nuevo y utilizarlo para la llamada.
- **Refresh token:** este token suele tener una vida útil bastante mayor, en nuestro sistema le asignamos 300 días. Cuando la vida útil del refresh token se acaba, el usuario deberá autenticarse nuevamente con su mail y contraseña, debido a esto es que se le suele asignar una vida útil más grande.

6.2.4. PostgresSQL

Decidimos utilizar PostgresSQL (`https://www.postgresql.org/`), una base de datos de código libre, relacional que usa y extiende el lenguaje SQL, combinado con varias funcionalidades que persisten los datos de forma segura y escalable. Esta base de datos ha ganado una reputación muy fuerte por su arquitectura probada, confiabilidad, integridad de datos, conjunto de características robustas, extensibilidad, y además contar con una comunidad dedicada al código abierto, que mejoran el rendimiento y proponen soluciones innovadoras.

PostgreSQL incluye varias funcionalidades pensadas para ayudar a los desarrolladores a construir aplicaciones, administradores para proteger la integridad de los datos y ayuda para manejarlos no importa si se cuenta con un conjunto de datos pequeño o grande. Además de ser de código abierto, es ampliamente extensible, por ejemplo permite definir tipos de datos propios, construir funciones personalizadas y escribir código desde diferentes lenguajes de programación.

PostgreSQL cuenta con poderosas extensiones como la popular y conocida PostGIS (`https://postgis.net/`), extensión que decidimos usar en nuestro proyecto, dado que en un principio, teniendo en mente que una parte del sistema iba a necesitar persistir datos geográficos, nos pareció interesante hacer uso de esta extensión, que agrega soporte para objetos geográficos, habilitando consultas de ubicación en el lenguaje SQL. Además, agrega tipos de datos extra a la base de datos PostgreSQL, como geometría, geografía, raster, entre otros. También funciones, operadores y mejoras en los índices que son aplicados a estos tipos espaciales. Estas mejoras hacen que sea más rápido y robusto trabajar con tipos geográficos en el manejo de un

sistema de base de datos.

En específico, las funcionalidades que agrega son:

- Tipos geométricos para puntos, líneas, polígonos, multipuntos, multilíneas, multi polígonos y colecciones de geometría.
- Predicados espaciales para determinar la interacción de geometría.
- Operadores espaciales para determinar métricas geoespaciales como área, distancia, largo y perímetro.
- Operadores espaciales para determinar un conjunto de operaciones geoespaciales como unión, diferencia, diferencia simétrica y buffers.
- Mejora en los índices para una mayor velocidad, rendimiento en las consultas y reducir memoria.

6.2.5. Heroku

Heroku (`https://www.heroku.com/`) es un servicio de plataforma en la nube (PaaS por sus siglas en inglés Platform as a Service), que permite a los desarrolladores hacer deploy y manejar sus aplicaciones de una manera fácil y amigable. Es una plataforma completamente administrada, por lo que los desarrolladores pueden concentrarse en su producto, sin tener que ocuparse del mantenimiento de servidores, hardware, o infraestructura. Provee de servidores, herramientas, servicios, y todo lo necesario para dejar una aplicación funcionando en determinada URL.

Es así, que creamos dos ambientes en esta plataforma:

- Staging: pensado para liberar versiones que serían probadas por el resto del equipo de MIA. Este ambiente sería muy similar a producción pero con acceso solamente por miembros internos, cada cambio nuevo es liberado en staging para ser probado antes de liberar la versión final a producción, incluido el rendimiento del servidor final, motivo por el cual, este ambiente luego fue cambiado al servidor de Facultad de Ciencias, pero nos sirvió para en una primera instancia, obtener retroalmientación temprana de las funcionalidades de la aplicación, sobre todo en el área palinológica.
- Develop: ambiente interno solamente entre desarrolladores, en el cual se trabaja en el desarrollo del día a día, realizando las pruebas pertinentes antes de liberar versiones al resto del equipo. Este ambiente se mantiene hoy en día en Heroku, no vimos la necesidad de cambiarlo siendo que tiene algunas funcionalidades útiles, como el deploy automático cada vez que que se suben cambios a la rama Develop, por lo cuál nos pareció útil seguir usándolo internamente.

Utilizamos un plan gratis destinado para aplicaciones no comerciales, pensado para pruebas de concepto, MVP's y proyectos personales. El cual cuenta, con 512MB de RAM, deploy aumático desde Github y logs.

Las limitaciones de este plan, es que el servidor se duerme luego de 30 minutos sin actividad, la primer request pasado este tiempo es quien despierta nuevamente al servidor, por lo que suele demora unos segundos. Además, mientras el servidor está despierto, está consumiendo horas, el plan gratis tiene un máximo de 550 horas por mes, lo cual es bastante más de lo que necesitábamos.

6.2.6. Envío de mails

En vista de la necesidad para el envío de mails de confirmación de cuenta, y resetear la contraseña, investigamos posibles servicios que pudieran ser utilizados por nuestra tecnología. Nos encontramos entonces con SendGrid (`https://sendgrid.com/`), utilizado por una gran comunidad de desarrolladores y grandes marcas como Uber, Spotify, airbnb, entre otras. Provee de un manejo de mails simple, donde agregando la clave asociada a la cuenta, se pueden enviar estos mails de forma sencilla.

Utilizamos el plan gratis, el cual incluye 100 mails por día, cantidad suficiente para la cantidad de usuarios que teníamos en vista para los siguientes meses. Uno de los principales problemas de este plan que no supimos identificar en su momento, motivo por el cual más tarde cambiaríamos de servicio, es el no contar con direcciones IP dedicadas. Esto implica que nuestra aplicación enviaría los mails desde una pull de direcciones IP, que probablemente estarían siendo usadas por otras aplicaciones. El problema radica cuando algún dominio de mails tenía bloqueada la IP con la que MIA intentaba hacer el envío, debido a que otra aplicación había usado la misma dirección para mandar spam. Esto derivó en que algunos usuarios no recibieran su mail de confirmación de cuenta, por lo que no podían confirmarla, y no podían acceder a la aplicación.

Para ese entonces, MIA ya se encontraba en producción para Android, por lo que nos vimos en la necesidad de buscar otra solución para este servicio, los sustitutos que encontramos fueron los siguientes:

- Servicio de mails que proveen los servidores de la Facultad de Ciencias.
- Gmail SMTP, el cual consta de crear un correo Gmail y usarlo para enviar mails automáticamente desde el backend.

Necesitábamos la opción que fuera más rápida posible, ya que era un error bastante crítico que podría seguir sucediendo a usuarios en producción, por este motivo, nos inclinamos por usar el servicio de Google y asegurarnos de liberar este arreglo rápidamente. Hasta el momento es el servicio que continuamos usando en los ambientes de producción y staging.

6.3. Servidor

Para hacer pública la aplicación era necesario un servidor web, el cual no podía estar ubicado fuera del país como lo están muchos servidores que ofrecen servicios

en la nube, dado que el sistema trabaja con datos personales del usuario, y por la ley de datos personales, estos solo pueden estar ubicados dentro del país. Es de aquí que surge la posibilidad de hostear el proyecto en un servidor de la Facultad de Ciencias. Para esto el equipo de informática de la Facultad de Ciencias nos generó una máquina virtual y un usuario usuario con permisos de administrador, a esta máquina virtual podemos acceder mediante SSH usando las credenciales provistas y la IP donde se encuentra la misma, siendo que el acceso es por SSH usamos el puerto 22.

6.3.1. Docker

Docker (`https://www.docker.com/`) es una plataforma de código abierto, el mismo permite a los desarrolladores empaquetar en contenedores, aplicaciones, componentes ejecutables estandarizados que combinan el código fuente de la aplicación con todas las bibliotecas y dependencias del sistema operativo necesarias (SO) para ejecutar el código en cualquier entorno.

Si bien se pueden crear contenedores sin Docker, este hace que sea más fácil, sencillo y seguro construir, implementar y administrar, es decir que permite crear, implementar, ejecutar, actualizar y detener contenedores mediante comandos simples y automatización que ahorra trabajo.

¿Por qué utilizar contenedores?
Los contenedores son posibles gracias al aislamiento y la virtualización de procesos del sistema operativo, que permiten que varios componentes de la aplicación comparten el kernel del sistema operativo subyacente.

Los contenedores ofrecen todos los beneficios de las máquinas virtuales, la escalabilidad rentable y la disponibilidad. Sin embargo, y aunque las máquinas virtuales mantienen las aplicaciones en el mismo hardware completamente separadas y reducen al mínimo los conflictos entre los componentes de software y la competencia por los recursos de hardware, estas son voluminosas (cada una requiere su propio sistema operativo, por lo que normalmente tienen un gran tamaño) y son difíciles de mantener y actualizar. En cambio en los contenedores, la capa adicional de abstracción (a nivel del sistema operativo) ofrece importantes ventajas adicionales:

- **Peso más ligero:** a diferencia de las máquinas virtuales, los contenedores no llevan la carga útil de una instancia de SO completa, solo incluyen los procesos y las dependencias del SO necesarios para ejecutar el código.

- **Mayor eficiencia de recursos:** con los contenedores, puede ejecutar varias veces la cantidad de copias de una aplicación en el mismo hardware que con las máquinas virtuales. Esto puede reducir su gasto en la nube.

- **Productividad mejorada del desarrollador:** en comparación con las máquinas virtuales, los contenedores son más rápidos y fáciles de implementar, aprovisionar y reiniciar. Esto los hace ideales para su uso en canalizaciones de integración continua y entrega continua (CI/CD) y un mejor ajuste para los equipos de desarrollo que adoptan prácticas Agile y DevOps .

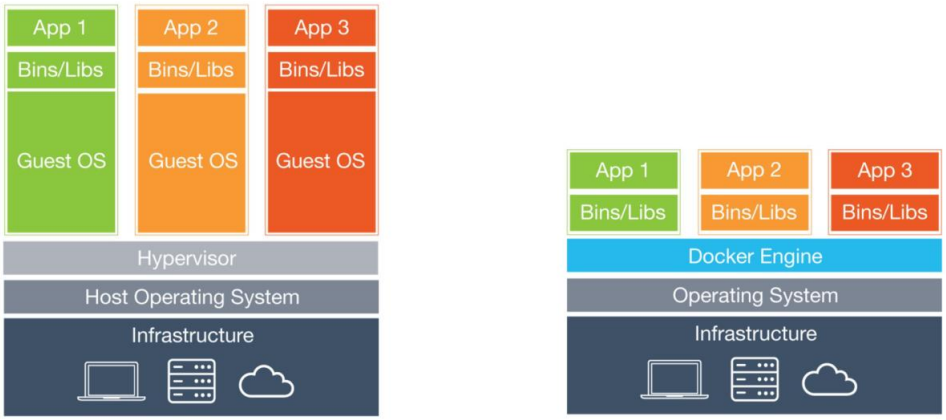


Figura 6.3: Maquina Virtual vs Docker

Docker Compose

Docker cli se usa cuando se utilizan contenedores individuales, por ejemplo en nuestro caso para correr nuestra página web, sin embargo para correr el backend utilizamos docker-compose, dado que este permite administrar una aplicación de varios contenedores, y nosotros teníamos dos, uno para la base de datos y otro para el backend Django .

Por todo lo que mencionamos anteriormente y aunque sabíamos que aprender a usar esta tecnología iba a requerir algo de tiempo, fue que decidimos usar Docker. Para hacer disponible la API REST creada en Django, y como ya mencionamos antes utilizamos docker-compose, fue necesario configurar un archivo Dockerfile, en el cual se indican todas las librerías y dependencias que eran necesarias instalar o correr, y un archivo .yml, en el que se indicaron los dos contenedores a usar, el primero de ellos configuraba la base de datos, para esto hicimos uso de una imagen ya existente *mdillon/postgis:9.5*. Esta imagen proporciona un contenedor Docker que ejecuta Postgres con PostGIS 2.5 instalado. Esta imagen se basa en la postgres imagen oficial y proporciona variantes para cada versión de Postgres 9 compatible con la imagen base (9.4-9.6), Postgres 10 y Postgres 11.

Esta imagen asegura que la base de datos predeterminada creada por la postgres imagen principal tendrá instaladas las siguientes extensiones:

- postgis
- postgis_topology
- fuzzystrmatch
- postgis_tiger_geocoder

El segundo de los contenedores usados fue el que indicaba cómo correr la aplicación Django. Configurar estos archivos así como instalar Docker y docker-compose en el servidor, fue todo lo que tuvimos que hacer para correr el proyecto aquí.

Por otro lado, y como ya mencionamos antes, para correr la página web, usamos Docker, con lo que configuramos únicamente el archivo DockerFile, en el mismo se

indicó que la imagen a usar debía a ser nginx [**nginx**]. Este es un servidor proxy inverso de código abierto para los protocolos HTTP, HTTPS, SMTP, POP3 e IMAP, así como un equilibrador de carga, caché HTTP y un servidor web (servidor de origen). El proyecto nginx comenzó con un fuerte enfoque en alta concurrencia, alto rendimiento y bajo uso de memoria.

A modo de conclusión, el uso de docker nos permitió correr las aplicaciones en el servidor, sin necesidad de instalar y configurar todo el software necesario para que las mismas corrieran.

6.3.2. Apache

Apache es un software de servidor web gratuito y de código abierto que alimenta alrededor del 40 % de los sitios web de todo el mundo. El nombre oficial es Apache HTTP Server (`https://httpd.apache.org/`), y es mantenido y desarrollado por Apache Software Foundation. Apache es rápido, confiable y seguro. Puede personalizarse en gran medida para satisfacer las necesidades de muchos entornos diferentes mediante el uso de extensiones y módulos.

¿Qué es un servidor web?

Los servidores de archivos, los servidores de bases de datos, los servidores de correo y los servidores web utilizan diferentes tipos de software de servidor. Cada una de estas aplicaciones puede acceder a archivos almacenados en un servidor físico y utilizarlos para diversos fines.

El trabajo de un servidor web es servir sitios web en internet. Para lograr ese objetivo, actúa como intermediario entre el servidor y las máquinas cliente. Extrae contenido del servidor en cada solicitud de usuario y lo envía a la web.

El mayor desafío de un servidor web es atender a muchos usuarios web diferentes al mismo tiempo, cada uno de los cuales solicita páginas diferentes. Los servidores web procesan archivos escritos en diferentes lenguajes de programación como PHP, Python, Java y otros.

Los convierten en archivos HTML estáticos y los sirven en el navegador de los usuarios web. Se conocen como la herramienta responsable de la comunicación adecuada entre el servidor y el cliente.

¿Cómo funciona el servidor web Apache?

Aunque llamamos a Apache un servidor web, no es un servidor físico, sino un software que se ejecuta en un servidor. Su trabajo es establecer una conexión entre un servidor y los navegadores de los visitantes del sitio web (Firefox, Google Chrome, Safari, etc.) mientras se entregan archivos entre ellos (estructura cliente-servidor). Apache es un software multiplataforma, por lo que funciona tanto en servidores Unix como Windows.

Cuando un visitante desea cargar una página en su sitio web, su navegador envía una solicitud a su servidor y Apache devuelve una respuesta con todos los archivos solicitados (texto, imágenes, etc). El servidor y el cliente se comunican a través del protocolo HTTP, y el software Apache es responsable de la comunicación fluida y segura entre las dos máquinas.

Apache es altamente personalizable, ya que tiene una estructura basada en módulos. Los módulos permiten a los administradores del servidor activar y desactivar funcionalidades adicionales. Tiene módulos de seguridad, almacenamiento en caché, reescritura de URL, autenticación de contraseña y más.

6.3.3. Certbot

Certbot (`https://certbot.eff.org/`) es una herramienta de software gratuita y de código abierto para generar automáticamente certificados Let's Encrypt (`https://letsencrypt.org/`) en sitios web administrados manualmente para habilitar HTTPS.

Certbot está creado por Electronic Frontier Foundation (EFF), una organización sin fines de lucro 501 (c) 3 con sede en San Francisco, CA, que defiende la privacidad digital, la libertad de expresión y la innovación.

HTTPS utiliza una combinación de dos protocolos (HTTP+SSL/TLS) que hace que cualquier tipo de información que se transmita en la red sea cifrada y nadie pueda acceder a ella, únicamente navegador y servidor web. Y para ello es necesario que la web tenga instalado un Certificado SSL. Fue entonces Certbot quien nos permitió certificar nuestros sitios web. El certificado SSL de seguridad es el encargado de cifrar o encriptar las conexiones entre el navegador/aplicación y servidor web, impidiendo que nadie pueda interceptar la información que se transfiere entre ambos. De este modo, todos los datos personales, bancarios o cualquier otro tipo de información sensible que se intercambie estará protegida.

6.4. Página web

6.4.1. Angular

Angular Angular es un framework JavaScript, gratuito y Open Source, creado por Google y por una comunidad de individuos y corporaciones, destinado a facilitar la creación de aplicaciones web modernas de tipo SPA (Single Page Application).

Este framework facilita el diseño y codificación de aplicaciones web de mediana y alta complejidad.

Angular utiliza el lenguaje Typescript, que es compatible con cualquier sistema operativo y navegador web. Además permite que su código pueda ser validado y corregido de manera eficiente ahorrando tiempo y trabajo al programador.

Entonces este framework es una plataforma muy efectiva para crear aplicaciones web, debido a:

- Angular tiene arquitectura MVC (Model, Template, View) que simplifica las líneas de codificación, lo que permite la creación de una aplicación más liviana y estructurada.

- Su estructura está basada en componentes y gracias a esto, hace que los mismos sean altamente reutilizables para construir cualquier aplicación, ya sea móvil o cliente-servidor, además de que estaremos creando aplicaciones web escalables.

- Este framework incorpora el lenguaje TypeScript que ofrece una serie de funcionalidades que garantizan una mayor versatilidad y robustez al momento de desarrollar aplicaciones web. Cualquier código desarrollado en este lenguaje, una vez compilado, se traduce en código JavaScript. Por ende tienes la posibilidad de seguir utilizando las librerías más conocidas para JavaScript, como por ejemplo jQuery.

6.5. Otras herramientas

6.5.1. Git y GitHub

Git (`https://git-scm.com/`) es un sistema de control de versiones distribuido de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia.

GitHub es una plataforma de alojamiento de código para el control de versiones y la colaboración. Permite a los usuarios trabajar juntos en proyectos desde cualquier lugar.

En particular utilizamos 3 repositorios para alojar el código: del backend, de la aplicación y de la página web. La metodología de trabajo para subir cambios a cualquiera de los repositorios, fue primero crear una nueva rana, en la cual se subirían los cambios correspondientes a una tarea de trabajo. La ramificación es la forma de trabajar en diferentes versiones de un repositorio a la vez.

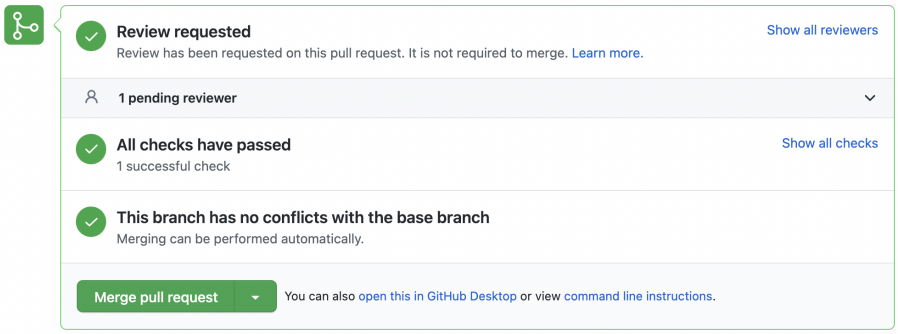
De forma predeterminada, un repositorio tiene una rama nombrada main que se considera la rama definitiva. Se usan ramas para experimentar y hacer ediciones antes de comprometerlas. Luego para agregar los cambios a las rama principal, creamos la regla de realizar un Pull Request (PR). Un PR es una solicitud enviada a un repositorio de GitHub para fusionar código en ese proyecto. El PR permite a los revisores ver y discutir el código propuesto. Una vez que el PR pasa todos los estándares de revisiones y se han realizado todas las revisiones necesarias, se puede fusionar con el código base. Los PRs son importantes porque ayudan a garantizar que el código revisado de calidad se combine en los repositorios de GitHub.

El código de MIA se encuentra de forma pública en `https://github.com/mialergia` donde se pueden encontrar los repositorios de la aplicación móvil, la página web y el backend.

6.5.2. CircleCi

CircleCi (`https://circleci.com/`) es una plataforma moderna de integración y deploy continuo (CI/CD), se utiliza como solución para correr automáticamente tests, builds, y deploys. Cuenta con un plan gratis que permite correr un proceso a la vez, 2500 créditos por semana, y puede ser corrido en Linux o Windows.

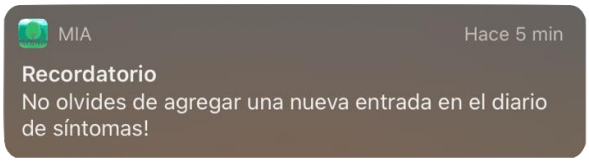
Decidimos utilizarlo integrado con GitHub tanto para el frontend como para el backend, para que corra el linter de la tecnología y los tests automatizados (este último sólo en backend), de esta manera, cada vez que se crea un commit, este es procesado por CircleCi y aparecerá con un tick verde si no hubo ningún error. De la misma manera, cada vez que sea crea un pull request, aparecerá un tick o una cruz en caso de errores, lo cual nos garantiza mantener la calidad de código y la tranquilidad de que los tests automatizados no se hayan roto con nuevos cambios.



6.5.3. OneSignal

OneSignal (`https://onesignal.com/`) es es un servicio de mensajería para aplicaciones móvil y web ofreciendo servicios de notificaciones push, mensajes in-app, SMS, mails, A/B testing, analytics, entre otros. Provee de una API abierta para manejar los mensajes del lado del backend, documentación extensa, y soporte para diferentes tecnologías de frontend, en particular para React Native que era la nuestra en específico. Utilizamos el plan gratis que permite, para mobile, un número ilimitado de miembros subscritos a las notificaciones.

Con esta herramienta, pudimos ofrecer a los usuarios la posibilidad de suscribirse a ciertos eventos, como avisarles cuándo un nuevo reporte se haya subido, o recordar una vez al día de subir una entrada en el diario de síntomas. Con la posibilidad de habilitar o deshabilitar la notificación en el momento que deseen. Además, podemos acceder a la web de OneSignal para poder obtener métricas de las notificaciones enviadas.



Capítulo 7

Pruebas y validaciones

Desde el principio del proyecto vimos la importancia de realizar tests automatizados, al menos del lado del backend, para asegurar la consistencias de los datos, que cada endpoint funcione como es esperado, y poder mergear nuevos cambios con la tranquilidad de que las funcionalidades desarrolladas anteriormente sigan funcionando como es esperado.

En un principio comenzamos realizando cada funcionalidad junto con su respectivo test, pero más avanzado el desarrollo nos vimos en la necesidad de dejarlos de lado y seguir solamente con el desarrollo de funcionalidades, por temas de tiempo. Una vez terminado el alcance principal del proyecto, dedicamos el tiempo necesario para agregar los tests que faltaban, para así seguir con el ritmo que teníamos en mente: acompañar cada funcionalidad nueva con su respectivo test.

Realizamos hasta el momento 35 tests automatizados, que incluyen, principalmente, para todos los endpoints usados por el usuario, el obtener la lista, crear y eliminar las instancias del modelo, además de otras funcionalidades específicas que queríamos probar en casos puntuales. Tenemos un porcentaje de cubrimiento del 90 %.

Como se comentó anteriormente, utilizamos la herramienta CircleCi para correr estos tests de forma automática cada vez que se crea un commit o un pull request, de forma de asegurarnos de no mergear código nuevo que rompa alguno de los tests desarrollados anteriormente. Por otro lado, también agregamos la regla de mostrar un error en caso de que el porcentaje de cubrimiento de todos los tests no supere el 85 %, con lo cual en un futuro si se siguen agregando funcionalidades pero no sus tests, el porcentaje de cubrimiento comenzará a bajar y CircleCi lo advertirá con un error.

7.1. Factories

Utilizamos la librería Factory Boy (`https://factoryboy.readthedocs.io/`) para simular instancias de los modelos con datos falsos. Estos se declaran con el tipo de datos que requiera (textos, párrafos, mails, fechas, etc.) y luego se podrán crear instancias de estos modelos.

Permite desligarse de crear datos estáticos y a mano que puedan estar sesgados por el programador, y usar en cambio datos creados totalmente al azar, con ciertas customizaciones como el largo, el formato, entre otros. Además, cuenta con soporte para todas las relaciones de manera de crear modelos exactamente iguales a los reales: many-to-many, many-to-one, one-to-one.

7.2. Tests

A continuación se detallan los tests creados hasta el momento, cabe destacar que por cada test que se corra, la base de datos comenzará vacía. Además, existen test generales que suelen probar las mismas características:

- Crear satisfactoriamente: se testea el caso satisfactorio, donde se realiza la request con los datos esperados, se comprueba que la respuesta sea 201 (CREATED), que los datos devueltos sean los mismos que los ingresados, y que la cantidad de elementos en la base de datos para ese modelo sea uno.
- Obtener: se testea obtener la lista de elementos creados, se comprueba que el status sea 200 (OK), y que los datos devueltos sean los mismos que se encuentran en la base de datos para ese modelo.
- Eliminar: se testea eliminar un elemento, se comprueba que la respuesta sea 204 (NO CONTENT), y que la cantidad de elementos en la base de datos para ese modelo sea vacío.
- Validación: se testea crear con algún dato erróneo, y comprobar que la respuesta sea 400 (BAD REQUEST), que no se haya creado la instancia y que devuelva un mensaje de error.
- Autenticación: se testea al intentar utilizar un endpoint sin un usuario autenticado, entonces la respuesta será 401 (UNAUTHORIZED).
- Autorización: se crean datos con un usuario logueado, se lo desloguea y se loguea nuevamente con un usuario distinto, al consumir el GET de un endpoint, la respuesta es satisfactoria pero los datos son vacíos, dado que el usuario autor de los datos no es el autenticado actualmente.

7.2.1. Usuarios

Crear una cuenta

- Crear satisfactoriamente.
- Con un mail que ya está en uso (validación).
- Con contraseñas que no coinciden (validación).

Iniciar sesión

- Crear satisfactoriamente.
- Con un usuario que no existe (validación).

Actualizar los datos

- Autenticación
- Crear satisfactoriamente.
- Sin nombre (validación).
- Sin departamento (validación).
- Sin barrio:
  Comprobar que si el barrio es Montevideo y no se indica el barrio, entonces un mensaje de error es devuelto indicando que el barrio no puede ser vacío (en caso que el departamento sea diferente a Montevideo, no se pide el barrio).
- Con un identificador de notificación incorrecto:
  Comprobar que si la notificación agregada no existe en la base de datos, se devuelve un mensaje de error.

Refrescar el token

- Refrescar satisfactoriamente.
- Refrescar con un token vació y asegurarse que la respuesta sea 401 (UNAUTHORIZED), dado que cuando el frontend recibe esta respuesta, desloguea automáticamente al usuario.

7.2.2. Síntomas

Historial

- Autenticación
- Autorización
- Crear satisfactoriamente con síntomas.
- Crear satisfactoriamente sin síntomas agregados.
- Obtener.
- Eliminar.

Diario de síntomas

- Autenticación
- Autorización

- Crear satisfactoriamente.
- Crear una entrada con una o dos respuestas críticas, y comprobar que una alerta de gravedad baja es devuelta: “Te sugerimos iniciar el tratamiento recetado e ingresar tus síntomas en el diario al menos una vez al día, por una semana.”.
- Crear una entrada con tres o cuatro respuestas críticas, y comprobar que una alerta de gravedad media es devuelta: “Te sugerimos iniciar el tratamiento recetado y consultar con tu médico.”.
- Crear 5 entradas en la misma semana con una respuesta crítica cada una, y comprobar que una alerta de gravedad alta es devuelta: “Por favor, agendá una consulta con tu médico.”.
- Obtener todas las entradas.
- Eliminar una entrada.

Preguntas, síntomas y categoría de los síntomas

- Autenticación
- Si bien estos endpoints no son usados directamente por el usuario, se crearon con la idea que, quizás en un futuro, puedan ser usados por las médicas para poder crear nuevas preguntas o síntomas y éstos aparezcan automáticamente en la aplicación. Para estos endpoints se testó que se pueda crear, eliminar y obtener los elementos.

7.2.3. Polen

Reportes de concentración

- Obtener los reportes
- Obtener el detalle de los reportes

Capítulo 8

Trabajos futuros

8.1. Nuevas funcionalidades

Dado el éxito de la aplicación, el numeroso alcance de usuarios y el buen desarrollo del trabajo interdisciplinario, estamos en la búsqueda de financiamiento para poder seguir manteniendo lo que ya tenemos, y poder crear una nueva versión con nuevas funcionalidades. En esta sección se enumeran aquellas funcionalidades que pensamos podrían ser de interés para agregar en esta nueva versión.

8.1.1. Datos geográficos

Cruzar los datos palinológicos con datos geográficos era una de las funcionalidades que se tenía en mente al principio del proyecto y finalmente debió desplazarse para un futuro. Teniendo en cuenta que aún vemos la necesidad y la gran ayuda que estos datos tendrían para el usuario, es una de las funcionalidades que se busca agregar en una próxima versión.

La idea principal se basa en agregar una nueva pestaña en la aplicación donde los usuarios puedan visualizar el arbolado público de Montevideo en una especie de mapa de calor. Nuestro sistema se encargaría de cruzar los datos de aquellas concentraciones de pólenes que sean críticas, con la ubicación de los árboles que las afectan, y de esta manera mostrar con colores en el mapa aquellas zonas que podrían ser críticas para sus alergias. De esta forma el usuario podría visualizar aquellas zonas que podrían afectarlo, y decidir su recorrido en base a estos datos.

Actualmente contamos con el censo de arbolado en Montevideo del año 2008 [5], son datos que pueden ser accedidos públicamente y descargarse en formato csv, nuestro trabajo radica en cargar estos datos en la base y procesarlos de manera de saber cuánto afecta cada uno de ellos a las alergias de los usuarios, así como filtrar aquellos tipos arbóreos que no tienen ninguna incidencia en las alergias, una vez tengamos estos datos se podría dar comienzo a la visualización del mapa del lado del frontend.

8.1.2. Medicamentos

En algunas reuniones con el equipo de medicina se discutió sobre darle la posibilidad a los usuarios de ingresar en el sistema los medicamentos que toman como parte de su tratamiento, esta fue una de las funcionalidades que no se logró llevar a cabo, dado que al momento de priorizarlas se le asignó una prioridad baja. Esta funcionalidad consta de tener una lista de medicamento en el sistema con su nombre científico, así como con sus nombres comunes, dado que muchas veces existen varios medicamentos con la misma droga pero que son producidos por distintas farmacéuticas.

El registro de medicamento pretendía que los usuarios no olvidaran qué medicamentos les recetó su médico, pudiendo agregar también información de cuándo tomarlo, la dosificación y dándole la opción de crear alarmas para no olvidarlo. La lista de medicamento también se pretendía mostrar a la hora de ingresar un nuevo evento en el diario de síntomas, con el fin de que el usuario seleccionara los medicamentos que consumió, si eran parte de su tratamiento o no, o en el mejor de los casos, simplemente indicar que no tomó ninguno.

8.1.3. Estado del tiempo y contaminación en el aire

La rinitis alérgica en los paciente se puede ver afectada por el polen y esporas de hongos de árboles, arbustos, pastos y hongos, pero estas no son las únicas causantes para que la rinitis alérgica se ve afectada. Se sabe que el desencadenamiento de nuevos síntomas en los pacientes pueden estar ligados al estado del tiempo y/o a la contaminación en el aire, un estado del tiempo con alta humedad o alta concentración de contaminación en el aire pueden ser tan nocivos como el polen y esporas de hongos. Es por esto que es deseado contar en la aplicación con estos datos pera que el usuario también pueda tomar las precauciones necesaria, sin necesidad de consultar muchos sitios para hacerlo. Por un lado el estado del tiempo es accesible a través de distintas páginas web, y sabemos que existe sistemas que exponen mediante una API los datos del clima, por lo que consideramos que tanto para mostrarle a los usuarios los mismos, como para hacer un cruzamiento de datos con los que se cuenta en el sistema, sería relativamente sencillo. Por otro lado los datos de contaminación en el aire son registrados por la intendencia de Montevideo, pero no son expuestos inmediatamente, por lo tanto para exponer a los usuarios datos de este tipo requeriría un gran esfuerzo, empezando por coordinar con acciones con el grupo de la intendencia encargado de este accionar, y seguido de generar un sistema capaz de guardar estos datos de forma dinámica, para luego ser expuestos en la aplicación de modo de mostrarle al usuario datos actualizados y reales. Además estos datos no solo se mostrarían a los usuarios sino que sería de interés cruzarlos con los demás datos recolectados.

8.2. Tratamiento de datos recolectados

En esta sección se enumeran las distintas formas de utilizar y mostrar a los otros miembros del equipo los datos recolectados, tanto desde la aplicación como desde

el administrador, es decir, mostrar de distintas formas los datos ingresados por los usuarios y los datos recolectados por el equipo de palinología. Para esto se pensó en realizar una nueva página web, donde los usuarios tengan distintos roles, pudiendo acceder a distintos datos dependiendo del rol que se tenga, los roles en un principio serían medicina y palinología.

8.2.1. Visualización de datos recolectados

Hoy en día los datos recolectados, ya sea los datos palinológicos, como los datos ingresados en los diarios de síntomas, no son mostrados más que en la aplicación, la idea es mostrar todos los datos recolectados, principalmente en tablas y gráficas. Estos se cruzarían con fechas, mostrando por ejemplo los niveles de polen y esporas de hongo para períodos de tiempo. También se podría mostrar la cantidad de eventos en el diario que fueron ingresados en fechas o períodos de tiempo, así como también la gravedad de los eventos ingresados en el diario de síntomas. Por otro lado, es deseado cruzar datos palinológicos con los datos ingresados por los usuarios, con el fin de mostrar ante qué concentraciones de polen y esporas de hongos se sintieron peor.
El cruzamiento de estos datos de forma macro podría arrojar datos importantes para las ciencias que involucraron este proyecto, pudiendo obtener varias conclusiones de los mismos.

8.2.2. Seguimiento de los datos del usuario por parte de su médico

En adición a la sección anterior donde se mostrarían los datos de forma genérica y macro, aquí se plantea darle la habilidad al médico de poder ver los datos ingresado en el diario de síntomas por cada paciente de forma individual. Para que el médico tratante pueda acceder a los datos de sus pacientes, en necesario que los mismos den permiso a esta acción, es decir, que debería estar de acuerdo en que su médico tratante tenga acceso a los datos que ingresó en el sistema, y además debería indicar quién es su médico tratante. En otro caso el personal médico no tendría los permisos suficientes para poder ver los datos de su paciente, este filtro sería agregado de forma de cuidar los datos personales de cada usuario, y agregar un paso más de validación para poder compartir sus datos.
Así es como en la página web los médicos podrían ver todos los pacientes que indicaron que se atienden con ellos, en conjunto con los eventos que hayan ingresado en el diario. Esto le permitiría a los médicos, a la hora de tener consultas con sus pacientes, determinar cómo se han sentido en el último tiempo teniendo al alcance una forma rápida de ver los datos ingresados.

8.2.3. Exportación de datos recolectados

Hoy en día el equipo de palinólogas tienen la posibilidad de ingresar diariamente los reportes en el administrador, y estos datos son guardados en la base de datos, la carencia en este flujo radica en que no tienen una forma de poder ver y descargar

los reportes que han subido a lo largo del tiempo, por esto surge la necesidad y el pedido por parte de su equipo de tener la posibilidad de poder exportar estos datos en formato compatible con excel o similar. Sumado a esto, también se podría extender esta funcionalidad de exportación de datos para todos aquellos datos de cruzamiento que se agreguen en un futuro.

8.3. Mejoras generales

En esta sección se enumeran las posibles acciones a tomar para mejorar el estado actual del sistema.

8.3.1. Manejo de Errores

Una de nuestras prioridades para una segunda versión es tener una forma de poder identificar errores críticos en producción, ya sea del lado del backend o el frontend. Necesitamos poder integrar una herramienta que nos alerte cuando sucede algún fallo crítico en alguna de las funcionalidades de la aplicación, esto nos ayudaría a identificar aquellos errores que produzcan que la aplicación se cierre, o que afecten los pasos para terminar algún flujo.

Una herramienta bien conocida para estos casos y que cumple exactamente con nuestra necesidades es denominada Sentry (`https://sentry.io/`), creada para el rastreo y monitoreo de errores para más de 30 tecnologías, en particular, tiene soporte para React Native y Django , nuestras tecnologías en frontend y backend respectivamente. Integrando esta herramienta a nuestro sistema, podríamos monitorear en la plataforma de Sentry los errores que han tenido los usuarios al hacer uso de la aplicación, los pasos que siguieron hasta antes de que ocurriera el error, e incluso datos del dispositivo que estaban usando (sistema operativo, versión, etc). Por lo tanto no solo podríamos atacar problemas que ocurren en tiempo real, sino también identificar falencias del sistema que podamos atajar evitando futuros errores.

Sentry cuenta con un plan gratis el cual tiene una limitante por mes de, entre otros:

- 50 mil reportes de errores, los cuales son enviados cada vez que sucede algún error en el sistema (o también pueden enviarse manualmente).
- 100 mil transacciones para monitoreo de rendimiento, las cuales son enviadas cada vez que nuestro sistema envía o recibe una request y envía una response.

8.3.2. Rendimiento

En caso de que la cantidad de usuarios siga creciendo o que el uso de la aplicación por parte de los usuarios aumente (esto podría ocurrir si se agregan más funcionalidades a la aplicación de forma que los usuarios tengan más interacción con el

sistema), entonces habría que asegurarse de que el servidor responda en tiempo y forma.

Una de las tácticas altamente usada en la actualidad es la cremación de "granjas de servidores" o grupos de servidores, de esta forma el procesamiento de los pedidos al servidor estaría distribuido entre todos los servidores que tenga la "granja", aquí es importante que no se sobrecargue ningún servidor, para esto sería prudente contar con un balanceador de carga que distribuya eficientemente el tráfico de red entrante entre el grupo de servidores.

Cuando se crean estos grupos, y se mantiene una base de datos, el cuello de botella puede estar dado en el acceso a la misma, una solución a esto podría ser cachear información, sabemos por ejemplo, que los reportes se actualizan como máximo una vez al día, entonces esta información podría estar en memoria, evitando ir a la base cada vez que se consulta estos datos. Otra soluciones a los cuellos de botellas por acceso a la base de datos, son la replicación de datos en distintas bases, aquí hay que tener cuidado con la consistencia de los datos, es decir que todas las bases de datos tengan la misma información.

Otra técnica puede ser la de partición de la base, es decir que se distribuirían las tablas en distintas bases, esta técnica no tiene inconsistencias. La replicación tiene como desventaja el costo de mantener todas las tablas con distintos datos, pero tiene como ventaja que es más escalable y que agrega seguridad, ya que si algo ocurre con una de las bases de datos, su información no se perderá.

8.3.3. Seguridad

Hoy en día, los únicos datos sensibles son los datos ingresados por los usuarios en el diario de síntomas, los cuales refieren a datos de salud, estos datos se encuentran en una tabla separada de los datos personales del paciente, pero se puede vincular con el usuario mediante una clave foránea, aunque los mismos solo son mostrados a los usuarios autenticados y el acceso a la base de datos es restringido y solo el grupo de ingeniería puede acceder a estos, ante un ataque, un intruso podría conocer los datos sensibles de los pacientes, es por esto que una posible mejora en lo que respecta a seguridad, sería cifrar los datos datos sensibles o cifrar las clave foránea para que un intruso no pueda vincular los eventos en el diario de síntomas con el paciente.

Por otro lado, actualmente no contamos con ningún sistema de respaldo de los datos, es por esto que, una de las acciones que pensamos tomar en la brevedad es la de respaldar los mismos o los que sean considerados más importantes, para que, ante la existencia de un intruso malintencionado, o la ocurrencia de un incidente o falla, no perder información valiosa. Como solución a esto existen dos grandes opciones, crear una copia de la base de datos en el mismo servidor, en este caso no nos cubriríamos en totalidad, ya que ante un accidente fatal que afecte a todo el servidor de la Facultad de Ciencias, estos datos se perderían de todas formas, otra opción sería guardar una copia de los mismos en un servidor externo, para esto habría que determinar cómo y dónde se haría esta copia. Otra cuestión a definir es cada cuanto tiempo es querido que se respalden los datos, lo ideal sería hacerlo diariamente de forma automática.

Capítulo 9

Conclusiones

9.1. Trabajo interdisciplinario

El objetivo del proyecto semillero era dar un puntapié inicial al trabajo entre las diferentes áreas de la medicina, biología, palinología e ingeniería, se buscaba en un principio poder identificar y cuantificar los granos de polen y las esporas de hongos, lograr un intercambio fluido en la comunicación entre las tres áreas, así como también cruzar los datos palinológicos con los médicos, y poder dar a conocer esta información fuera del equipo, así es como nuestro sistema es usado para el procesamiento de los datos, y la aplicación para la visualización de los mismos.

La mayoría de los objetivos planteados en un principio fueron cumplidos exitosamente a lo largo del ciclo de vida del proyecto, la comunicación entre las diferentes áreas se concretó armoniosamente, en total se realizaron 6 reuniones entre todos los integrantes del equipo y una decena entre subgrupos. Debido a la situación de pandemia del momento, las reuniones se realizaron a través de la plataforma Zoom, así como también de la aplicación de mensajería WhatsApp para una comunicación más directa.

Se logró desarrollar y poner en producción en fecha un producto con un excelente potencial de alcance de usuarios, disponible para dos de los sistemas operativos móviles más usados mundialmente, dando visualización a los reportes diarios obtenidos por el equipo de palinología, pudiendo crear alertas para las alergias diagnosticadas por el equipo de médicas.

Trabajar en un proyecto en el transcurso de una pandemia sin lugar a dudas jugó un papel importante, como se mencionó anteriormente, se tenía la idea de que las médicas pudieran difundir la aplicación entre los pacientes de la policlínica de rinitis que se formaría en noviembre, a pesar de que la policlínica abrió, la concurrencia de los pacientes fue poca debido al marco de protocolos y distanciamiento social exhortados para ese momento, por lo cual se debieron buscar otros medios de difusión que ayudaran a cumplir con este objetivo, aquí es donde juegan un papel importante los medios de comunicación. Estos protocolos se vieron aplicados también en el área de la Intendencia Municipal de Montevideo, por lo que el objetivo de incluir los

espacios verdes en la aplicación también se vio truncado, motivo por el cual lo agregamos como un punto importante de trabajo a futuro.

9.2. Medios de comunicación

Figura 9.1: Prensa

La prensa ha sido un factor fundamental en cuanto a la llegada al usuario final, aprovechamos una iniciativa de la Facultad de Ciencias sobre difusión de investigaciones, nos pusimos en contacto con la encargada de esta iniciativa para contarle de nuestro proyecto. El tema fue de interés, por lo que quedamos en contacto con algunos productores, quienes solicitaron más información para poder enviarla a los medios de prensa.

Así surgió la primera publicación de MIA en el diario El Observador, en la sección especial para noticias de ciencia y tecnología, llamada Cromo. Se publicó en la web el día 5 de febrero del 2021 [6] y en el diario al día siguiente. El impacto de esta primera publicación fue muy bueno, fue el primer punto de aumento en el número de descargas. Además de ser una ventana de visibilidad para otros portales.

El siguiente medio de prensa interesado fue Radio Universal, del programa De taquito a la mañana, se entrevistó a Ángeles el 15 de febrero del 2021, se trató de una extensa charla de 18 minutos donde se habló sobre el contenido de la aplicación, cómo funcionan los reportes subidos, y contar sobre el proyecto y su equipo. El audio de la entrevista se puede encontrar en la página de la radio [7].

La tercer publicación fue el 20 de marzo del 2021 para Montevideo Portal [8], en esta nota en particular se agregó un enlace con acceso a la página web del proyecto, la cual cuenta con accesos directos a las tiendas donde la aplicación se encuentra publicada. Pensamos que esta debe ser la razón por la cual se vio un gran aumento

de usuarios luego de esta publicación.

Por último, se contactaron desde Telenoche (`https://www.telenoche.com.uy/`) para realizar una entrevista a todo el equipo, cada área en su espacio de trabajo, para poder emitirlo por televisión con motivo del día mundial de la alergia, el cual toma lugar el día 8 de Julio del 2021. El video final fue transmitido por Canal 4 el día 11 de Julio del 2021 al medio día y nuevamente en la noche, también fue publicado en su página web [9], teniendo el aumento de usuarios más grande desde que lanzamos la aplicación.

La aplicación también fue mencionada en el boletín de la Asociación Latinoamericana de Paleobotánica y Palinología. Está siendo recomendada en los tratamientos recetados por las médicas. Fue presentada ante a la Sociedad Uruguaya de alergia e inmunología.

A continuación se muestran dos gráficas para ambos sistemas operativos, Android y iOS, tomadas directamente de las plataformas donde se maneja el versionado y deploy de la aplicación, Google Play Console y App Store Connect, respectivamente.

Estas gráficas muestran el número de descargas entre los meses febrero y agosto del 2021. Como se puede ver, existen tres grandes picos en los rangos de fecha entre el 4 y 7 de febrero del 2021, el 19 y 22 de marzo del 2021, y el entre el 10 y 12 de Julio del 2021, fechas que coinciden con las publicaciones de El Observador, Montevideo Portal y Telenoche, respectivamente. Notar que, como se dijo anteriormente, el segundo pico es mayor que el primero, gracias a haber publicado un enlace a nuestra web donde los usuarios encontrarían dónde descargar la aplicación. Y el pico mayor en todo el rango de fechas se da luego de publicada la entrevista por Telenoche en televisión.

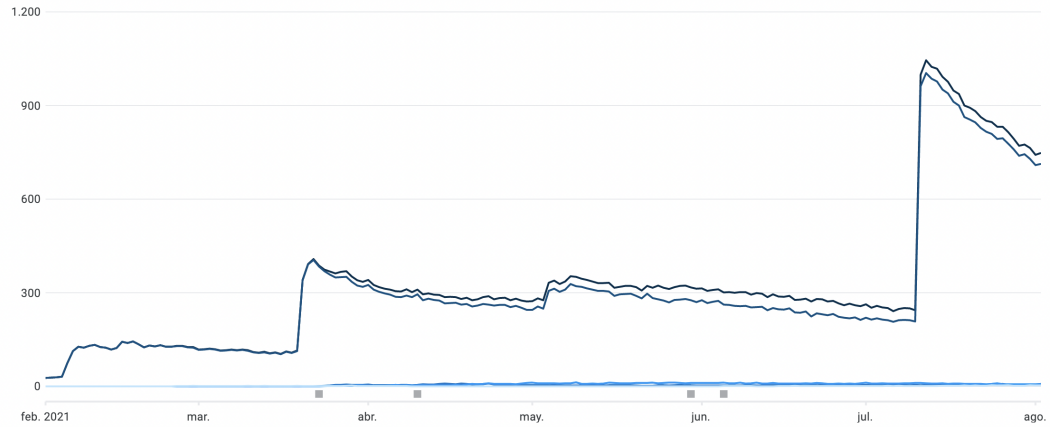


Figura 9.2: Usuarios Android con la aplicación descargada

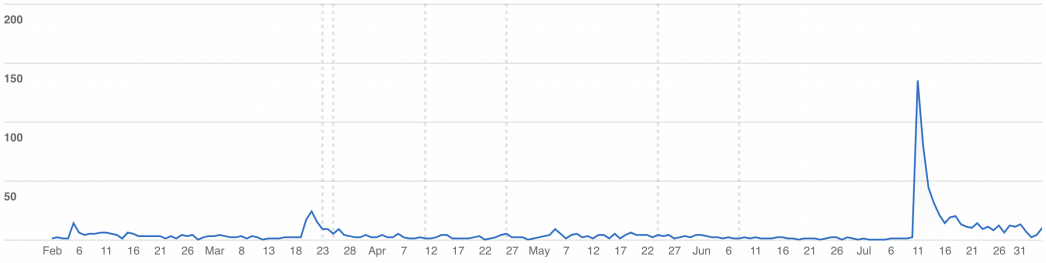


Figura 9.3: Usuarios iOS con la aplicación descargada

Tomando los números de Android que son más significativos, mientras la nota de El Observador aumentó la cantidad de instalaciones de 31 a 127, la de Montevideo Portal lo hizo de 115 a 408, y la entrevista de Telenoche aumentó de 244 a 1045. Como se puede observar además, si bien el número de instalaciones disminuye luego de cada pico de descargas, de todas formas el número total con respecto a antes de haber salido la nota, sigue representando un aumento en la cantidad de instalaciones activas.

Esto sugiere que la mayoría de los usuarios que han estado interesados en descargar la aplicación, encontraron una utilidad en MIA, aún así, dado la baja de números que se encuentra después de cada pico, se desprende que hay un porcentaje de usuarios que no encuentran lo que buscaba. Queda como trabajo a futuro investigar cuáles son estas falencias, y qué se necesita para colmar sus expectativas, si simplemente no eran parte del público objetivo, o hubo algo que buscaban del sistema que se pudo haber incluido y no fue priorizado correctamente.

Por otro lado, podemos ver que la cantidad de usuarios en Android es mucho mayor a la cantidad de usuarios en iOS, esto nos permite concluir que en caso de un recorte de presupuesto, se podría investigar qué tan prioritario sería mantener esta plataforma, teniendo en cuenta la cantidad de usuarios promedio que ha tenido la aplicación a lo largo del año. Poniendo en la balanza, el gasto de publicar la aplicación (99 dólares americanos al año) y la cantidad de descargas que podría darse en esta plataforma.

9.3. Retorno de los usuarios

El retorno de los usuarios ha sido un punto de entrada muy enriquecedor para el proyecto y para definir qué mejoras se podrían implementar en la aplicación. Los usuarios pueden hacer uso de los comentarios que se permite dejar en las tiendas de distribución, o dejar sus sugerencias en la encuesta que encuentran una vez iniciaron sesión en la aplicación.

Esta simple encuesta trata de solamente dos preguntas donde deben indicar qué tan útil les resulta la aplicación, y un campo abierto para dejar sugerencias para una próxima versión. Cuenta con más de 80 respuestas las cuales son en su mayoría positivas, se destaca y agradece el trabajo interdisciplinario, además de agradecer el hecho de que sea una aplicación gratuita, dado que hay otros diarios de síntomas similares pero que son pagos. Se encuentran algunas sugerencias interesantes que se

detallan a continuación.

Puntos a mejorar:

- Los comentarios y críticas se dividen entre aquellos usuarios que pudieron usar la aplicación y aquellos que no pudieron crearse la cuenta. Vemos que el flujo de crearse y confirmar la cuenta es un punto de falencia que se debe mejorar, ya que para algunos usuarios se hace difícil entender el hecho que deben confirmar su cuenta tocando en el enlace que se les envía por mail. Algunos usuarios han manifestado que no se pudieron crear la cuenta, otros intentan confirmar la cuenta respondiendo el mail. Estos suelen ser los usuarios que se ven más frustrados por no poder ni siquiera haber usado la aplicación.

- Otro error recurrente es que muchos usuarios no son conscientes que la aplicación trata alergias a polen y esporas de hongos, y se ven limitados al no ver otras alergias en el listado, como por ejemplo alergia al polvo. Además, otros manifiestan que los nombres de las alergias son demasiado técnicos, este problema radica en que se hizo la aplicación con la idea de que los usuarios tendrían hecho el Prick test, por lo que sabrían exactamente a qué son alérgicos, aún así se da la posibilidad de poder seguir sin seleccionar ninguna alergia. Tal vez sea necesario aconsejar a los usuarios que se asesoren con un profesional y que soliciten realizarse el test, así como también se podría dar mas información sobre las alergias.

- Se han repetido algunas preguntas sobre si las mediciones son solamente en Montevideo, a pesar de haber agregado en una de las últimas versiones la ubicación de Montevideo en la pantalla principal, quizás es un punto que debería estar más claro.

- Encontramos también problemas de usabilidad gracias a ciertos comentarios. Un usuario reportó que le gustaría que hubiesen alertas para recordarle que debe subir un reporte en el diario de síntomas, lo cual ya existe, lo que nos hace pensar que debería tener más visibilidad y no asumir que el usuario llegará a esa pantalla sin ayuda.

- Se han visto también comentarios sobre que les gustaría tener alguna vía de comunicación con el equipo médico, por ejemplo queriendo hacer preguntas específicas de sus dolencias.

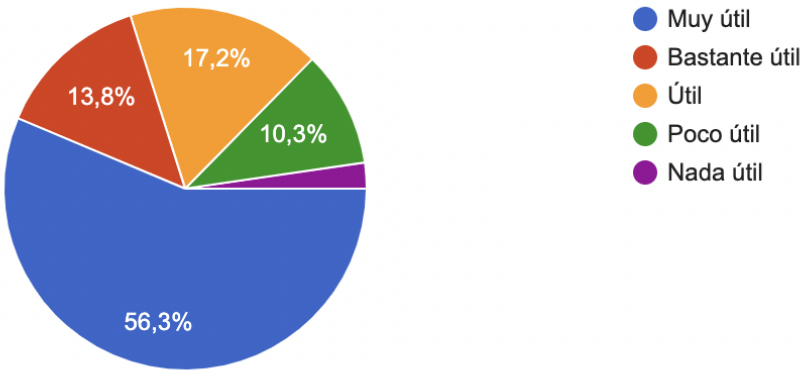


Figura 9.4: Respuestas a la pregunta: ¿Qué tan útil te resultó el proyecto?

9.4. Conclusión General

Sabemos que existen muchos puntos a mejorar, algunos de ellos diagnosticado a partir del retorno de los usuarios, lo cual es interesante, ya que de no haber tenido un producto tangible, con llegada a tantos usuarios estos puntos de mejora no hubiesen surgido. Por tanto consideramos estos puntos a mejorar como positivos y como desafíos para tener un producto de mayor interés y usabilidad para los usuarios.
Pese a la existencia de mejoras y nuevas funcionalidades, concediéramos que el producto entregado es sumamente valioso. Desde el punto de vista de la aplicación, los usuarios tienen acceso a datos de polen y esporas de hongo, datos no existentes en ningún otro sitio en el Uruguay, y no solo eso, los mismo son actualizados diariamente. Por otro lado, el diario de síntomas fue diseñado siguendo las recomendaciones de un equipo internacional conformado por los alergólogos/as más importantes del mundo, le da la posibilidad al usuario de tener un historial de sus síntomas, punto de sumo interés para aquellos que siguen un tratamiento, pudiendo lograr un tratamiento más adecuado.
Desde el punto de vista del administrador, este permite mostrar a los usuarios los datos recabados al instante ya que son guardados en base de datos, lugar mas fiable para contener esta información. El hecho de guardar estos datos en una base de datos, da un punto de partida para analizarlos, sacar conclusiones, realizar cruzamiento de datos, entre otras acciones deseados de interés para los integrantes del equipo.

Por otro lado, con lo que respecta al trabajo interdisciplinario y pese a no haber podido tener instancias presenciales, se logro una buena comunicación, pudiendo compartir y conocer otras áreas de la otras disciplinas, entendiendo que hacen y como es su forma de trabajo, lo cual consideramos que fue experiencia enriquecedora para todos los integrantes del equipo.

Con lo que respecta al producto entregado, concluimos que es un producto completo, dado que cuenta con una aplicación, un administrador y una pagina web, habiendo no solo dedicado tiempo al desarrollo de estas plataformas desde cero, sino también de estudio ya que no teníamos conocimiento previo en algunas de las herramientas utilizada.

Bibliografía

[1] Espacio Interdisciplinario. *Semillero de Iniciativas Interdisciplinarias*. URL: https://www.ei.udelar.edu.uy/programa-financiamiento/semillero-de-iniciativas-interdisciplinarias.

[2] Espacio Interdisciplinario. *Hiperreactividad respiratoria en Montevideo: Integración de factores ambientales para una medicina de precisión*. URL: https://www.ei.udelar.edu.uy/grupos-financiados/sigla-acronimo/hiperreactividad-respiratoria-montevideo-integracion-factores-ambientales-para-una-medicina.

[3] Yu-Feng Yvonne Chan y col. *The asthma mobile health study, smartphone data collected using ResearchKit*. Mayo de 2018. URL: https://www.nature.com/articles/sdata201896.

[4] M. Jones y col. *JSON Web Token (JWT)*. URL: https://tools.ietf.org/html/rfc7519.

[5] Intendencia de Montevideo. *Censo de arbolado 2008*. 2008. URL: https://catalogodatos.gub.uy/dataset/intendencia-montevideo-censo-de-arbolado-2008.

[6] El Observador. *Qué es MIA, la app uruguaya que busca ayudar a los alérgicos*. Mar. de 2021. URL: https://www.elobservador.com.uy/nota/que-es-mia-la-app-uruguaya-que-busca-ayudar-a-los-alergicos--20212594054.

[7] Vanessa Moreira Radio Universal. *App Mia: la aplicación que busca asistir al usuario alérgico en el seguimiento de sus síntomas*. Feb. de 2021. URL: https://970universal.com/2021/02/15/app-mia-la-aplicacion-que-busca-asistir-al-usuario-alergico-en-el-seguimiento-de-sus-sintomas/.

[8] Montevideo Portal. *MIA: la aplicación uruguaya que asiste a los usuarios alérgicos y brinda datos a médicos*. Mar. de 2021. URL: https://www.montevideo.com.uy/Ciencia-y-Tecnologia/MIA-la-aplicacion-uruguaya-que-asiste-a-los-usuarios-alergicos-y-brinda-datos-a-medicos-uc781287.

[9] Telenoche. *"MIA": La aplicación creada para ayudar a los alérgicos*. Jul. de 2021. URL: https://www.telenoche.com.uy/nacionales/mia-la-aplicacion-creada-para-ayudar-a-los-alergicos.

Appendices

Apéndice A

Conformación equipo interdisciplinario

El espacio interdisciplinario en el que trabajamos estuvo compuesto por áreas de la medicina, biología e ingeniería, quienes conforman estos grupos son:

Medicina

- Carina Almirón - Especialista en Otorrinolaringología
- Mariana Bonifacino - Especialista en Alergología
- Patricia Indarte - Especialista en Otorrinolaringología
- Rosario Eugui - Especialista en Otorrinolaringología

Por parte del grupo de médicas del Hospital Pasteur, su rol fue proveer todos los datos necesarios para lo que respecta a datos clínicos, dando una guía de qué datos eran de interés conocer de los pacientes, para así poder realizar tratamientos más específicos. Se encargaron de definir explícitamente las preguntas y respuestas que se les harían a los usuarios en cuanto a sus síntomas, además de definir la mejor terminología que se debería aplicar para que sea entendible por todo público.

Las médicas también contaban con presencia en la clínica Unidad Integral de Alergia (`https://uialergia.uy/`), que comenzaría en noviembre con duración de un mes, donde atenderían consultas, darían diagnósticos y tratamientos, lo cual marcó el deadline de nuestro proyecto, para poder así presentar la aplicación en dichas consultas y motivar su uso en pacientes diagnosticados.

Biología

- Ángeles Beri - Doctora en Ciencias Biológicas y responsable del proyecto
- Alejandra Leal Rodríguez - Doctorado en Ciencias Biológicas
- Ximena Martínez Blanco - Doctora en Ciencias Biológicas

Por parte del grupo conformado por docentes y palinólogas de la Facultad de Ciencias, su rol fue proveer de información de los valores de concentración de polen y esporas de hongos en el aire de Montevideo. Muestran aquellos potencialmente alergénicos, que son utilizados por el personal médico en los tests de alergenicidad (Pricks), estos muestreos son colectados en el barrio Malvín Norte, utilizando un colector de partículas ubicado a 12 m de altura. Además, calculan los niveles altos y medios de concentración, basándose en estudios realizados en Montevideo y en los niveles de concentración máximos definidos internacionalmente para cada tipo de polen o espora.

Ingeniería

- Libertad Tansini - Doctora en Ciencias de la Computación, co-responsable del proyecto Semillero y Tutora del proyecto de grado
- Regina Motz - Doctora en Ciencias de la Computación, Tutora del proyecto de grado
- Natalia Campiglia - Estudiante de Ingeniería en Computación
- Tamara Suárez - Estudiante de Ingeniería en Computación

Y por último, el grupo conformado por docentes y estudiantes de la Facultad de Ingeniería tuvo como rol la construcción de un sistema que pudiera concentrar tanto los datos de polen provisto por el grupo de palinólogas, así como hacer un seguimiento de los datos de los usuarios, para un mejor manejo de tratamiento por parte de las médicas. Notar que esto incluye, además de la construcción del sistema propiamente dicho, la recopilación de requisitos, determinación del alcance, prototipado, diseño y testing.

Apéndice B

Bitácora del desarrollo del Proyecto

B.1. Cronograma

A continuación se muestra un diagrama donde se puede ver gráficamente los tiempos y fechas de cada etapa del desarrollo de MIA. Se diferencian dos grandes fases dentro del proyecto.

B.1.1. Fase 1

Todo el trabajo relacionado a entender qué era lo que queríamos construir antes de empezar con la implementación. Esto incluye las primeras reuniones, estudio de trabajos relacionados, prototipos, crear, validar y priorizar casos de usos. Fue la etapa que nos llevó más tiempo debido al trabajo que requiere bajar a tierra cuando se tiene en mente una aplicación pero aún falta afinar los detalles, además en esta fase se necesitó mucho de reuniones entre todos los miembros, o reuniones separadas específicas con cada área, lo que también enlentece el proceso por la necesidad de coordinar una reunión que sirva para la mayoría de los participantes.

Del lado de medicina se discutió qué datos de los pacientes interesaba tener, qué funcionalidades les podrían ser útiles, cómo ayudarlos a reconocer patrones en sus síntomas. Del lado de la biología el desafío fue encontrar cómo mostrar los datos de la manera más amigable para el usuario, de forma que pudieran comprender fácilmente si los resultados de las mediciones eran o no favorables para sus patologías.

B.1.2. Fase 2

La segunda fase trata de la implementación del producto propiamente dicho. Antes de comenzar con el desarrollo realizamos un cronograma interno, teniendo en mente que el deadline debería ser los primeros días de noviembre, ya que en esta fecha comenzaría la atención en la policlínica y sería la oportunidad para recomendar a los pacientes que probaran la aplicación.

Marcamos fechas de finalización para los diferentes conjuntos de prioridades. Como se ve en el diagrama, las fechas estimadas fueron:

- **Alta prioridad:** 2 de setiembre
  - Crear cuenta
  - Iniciar sesión
  - Cambiar contraseña
  - Datos personales (onboarding)
  - Datos de síntomas (onboarding)
  - Mostrar datos de pólenes
  - Listado del diario de síntomas
  - Agregar evento al diario de síntomas
- **Media prioridad:** 29 de setiembre
  - Datos de alergias (onboarding)
  - Notificaciones
  - Detalle de los datos de pólenes
- **Baja prioridad:** 9 de octubre
  - Ver/editar datos de usuario
  - Agregar/editar alergias
  - Mostrar mapa
  - Agregar nuevo medicamento
  - Datos de medicamentos (onboarding)
  - Alerta de alergia personalizada
- **Testing:** 1 noviembre

Durante el proceso de desarrollo, algunas de estas prioridades se vieron cambiadas, y por eso se pude observar la diferencia en el diagrama entre la estimación y lo que fue la realidad.

Al llegar la fecha estimada de finalización de alta prioridad, el desarrollo de las funcionalidades en el backend y frontend estaba casi completo, no se pudo terminar a tiempo con el cambio de contraseña y la verificación vía email, aún así, pretendíamos liberar lo antes posible para poder recibir retroalimentación temprana.

Lamentablemente, no teníamos disponible en ese momento la configuración necesaria en el servidor de la Facultad de Ciencias, este fue un error al no haberlo tenido en cuenta en la estimación, si bien eramos conscientes que nos iba a llevar un tiempo prudencial, no identificamos que era la tecnología con la que estábamos

menos familiarizadas, y por tanto el área que más incertidumbre sobre tiempos se tenía. Por lo que terminamos liberando la primer versión en una fecha más tardía de los que hubiésemos querido, igualmente esto nos dio la posibilidad de incluir en esta primer versión una funcionalidad de media prioridad, que fue mostrar el detalle de los datos de pólenes, y además terminar con lo que no habíamos alcanzado: cambiar la contraseña y verificar la cuenta.

La primer versión de la aplicación fue muy útil, y fue de suma importancia haberlo hecha en una fecha temprana con respecto al deadline. Luego de liberada esa versión, tuvimos una reunión con las integrantes del área de biología, donde nos explicaron sobre algunos cambios que les gustaría añadir a la aplicación si diera el tiempo, donde explicaron que idealmente les gustaría ingresar en el administrador los valores medidos, y que el cálculo sobre qué tan críticos son se hiciera en nuestro sistema, con parámetros de niveles ingresados por ellas para cada tipo polínicos. Esto tenía un impacto en cuanto a los plazos que nos habíamos plantado ya que no lo habíamos tenido en cuenta.

A su vez, algunas necesidades cambiaron en cuanto a los requerimientos del proyecto. Para comenzar, se bajaron del alcance las funcionalidades sobre el mapa geográfico con los datos de la intendencia ya que no contábamos con los mismos, y además tampoco se veía la necesidad de aquellas relacionadas con los medicamentos, por lo que pudimos juntar las funcionalidades que faltaban de media prioridad junto con las de baja, para incluirlas todas en una segunda versión, teniendo todo el MVP completo, incluyendo los nuevos cambios pedidos por las palinólogas.

Es así que el 2 de octubre del 2020 estuvo disponible la primer versión de la aplicación en el Play Store, disponible abiertamente para todo público. La aplicación no estuvo disponible para iOS hasta el 2 de enero del 2021, debido a cuestiones burocráticas que se nos presentaron para la compra de la cuenta, si bien la aplicación ya funcionaba perfectamente en ambas plataformas, el pago de la cuenta nos llevó mucho tiempo por las comunicaciones con Apple, la Facultad de Ciencias y el tiempo que se toma Apple una vez pagada la cuenta para dejarla disponible.