INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

Desarrollo tecnológico y generación de riqueza sustentable

PROYECTO DE APLICACIÓN PROFESIONAL (PAP)

Programa de Ciudades Inteligentes

4L05 Vida Digital

Desarrollo móvil: Monitor Covid y WNS

PRESENTAN

Programas educativos y Estudiantes

Ing. Sistemas Computacionales. Isaac Cabrera Cortés

Ing. Sistemas Computacionales. Alessandro Pallaro Gómez

Profesor PAP: Mtro. Luis Eduardo Pérez Bernal

Tlaquepaque, Jalisco, agosto de 2020

ÍNDICE

Contenido

REPORTE PAP

Presentación Institucional de los Proyectos de Aplicación Profesional

Los Proyectos de Aplicación Profesional (PAP) son una modalidad educativa del ITESO en la que el estudiante aplica sus saberes y competencias socio-profesionales para el desarrollo de un proyecto que plantea soluciones a problemas de entornos reales. Su espíritu está dirigido para que el estudiante ejerza su profesión mediante una perspectiva ética y socialmente responsable.

A través de las actividades realizadas en el PAP, se acreditan el servicio social y la opción terminal. Así, en este reporte se documentan las actividades que tuvieron lugar durante el desarrollo del proyecto, sus incidencias en el entorno, y las reflexiones y aprendizajes profesionales que el estudiante desarrolló en el transcurso de su labor.

Resumen

En este PAP de Vida digital del periodo otoño 2020 se trabajó en el desarrollo móvil de una aplicación para monitorear e informar sobre los contagios de Covid en México. Este proyecto es la continuación del trabajo de alumnos del periodo de verano 2020, quienes aportaron el diseño de la aplicación y parte del *Backend*. Nosotros, el equipo de desarrollo móvil, utilizamos esos diseños para realizar la aplicación Monitor Covid para dispositivos Android y publicarla en la Play Store de Google para que esté disponible para todos.

Es este documento se registra todo el trabajo realizado a lo largo del semestre en el PAP Vida Digital. El objetivo de este reporte es evidenciar que cumplimos de forma efectiva el plan de trabajo y objetivos establecidos en las primeras semanas del PAP.En el documento se muestran evidencias de la aplicación y servicios web desarrollados a lo largo del semestre para poder hacer la liberación de la aplicación Monitor Covid.

1. Introducción

1.1. Objetivos

El primer objetivo del proyecto es ayudar a las personas a mantenerse informadas durante la pandemia. La forma en la que el PAP contribuirá a cumplir esto, es continuar con el proyecto propuesto en el periodo verano 2020 y terminar el desarrollo de la aplicación móvil **Monitor Covid**.

Por otro lado, se tiene el propósito de ayudar a la comunidad de la Zona Metropolitana de Guadalajara (ZMG) a monitorear el bosque de la primavera haciendo uso de la tecnología. Además de la utilización de drones para el monitoreo del bosque, se busca mantener informados a los interesados. Debido a esto, se desarrollará una aplicación móvil para que las personas interesadas puedan consultar los datos recolectados desde la palma de su mano.

1.2. Justificación

La aplicación móvil Monitor Covid tiene un gran peso en lo social para poder combatir a la pandemia del Covid 19 que hasta el momento ha afectado la salud de más de 25 millones de personas alrededor del mundo y ha causado más de 900 mil muertes. México es uno de los países que más casos presentan y los contagios aun no disminuyen. Además de las consecuencias que ha traído a la salud, la pandemia también ha afectado a la economía, es por lo que se debe hacer algo para terminar con este problema y poder volver a la normalidad lo antes posible. La aplicación que desarrollaremos puede ser una solución a este problema ya que al informar a la gente de las zonas donde está presente el virus se pueden evitar muchos contagios. La aplicación también servirá como una guía para aquellas personas que no conozcan como detectar y tratar la enfermedad puedan hacerlo fácilmente.

Por otro lado, la aplicación de la red de sensores ayudará a todos los investigadores del bosque de la primavera a poder recabar datos importantes sobre el estado del bosque. La

existencia del bosque es fundamental para que la ciudad pueda contar con aire limpio y para poder tener una buena calidad de vida.

En cuanto a la parte disciplinaria esto involucra totalmente temas vistos en la carrera, principalmente el desarrollo móvil y la administración de proyectos. La aplicación será desarrollada en Android nativo y utilizaremos metodologías agiles para trabajar de forma efectiva.

1.3 Antecedentes

A finales del 2019, en el mercado ahora mundialmente conocido Huanan, en Wuhan China, una persona contrajo el virus proveniente de un animal. Ese virus ahora conocido como Covid-19, afectó la vida de millones de personas.
Debido al mundo globalizado en el que vivimos, el virus no tardó mucho en salir de China y eventualmente alcanzar cada rincón del planeta. La rápida propagación del virus implicó hacer cuarentena en muchísimos países y poner en pausa actividades de todos los sectores económicos. La interrupción de todo tipo de actividades causó impactos en los sectores económico, social y político.
(Readfearn, 2020)

En los últimos 30 años de su existencia el bosque de la primavera ha perdido más patrimonio ecológico que en más de sus 140 mil años de su existencia. En el año pasado ocurrieron más de 50 incendios que dañaron más de 1493 hectáreas de bosque. Según especialistas, en el siglo pasado el daño que causaban los incendios al bosque eran menores a mil hectáreas por año. En 2005 se produjo un incendio de más de 11 mil hectáreas. El bosque de la primavera cuenta con 30 mil hectáreas de áreas protegida pero cada vez estas áreas han ido disminuyendo mientras que los inmobiliarios han ido aumentando y cada vez hay más propiedades en donde antes había bosques.

1.4. Contexto

Solo en México, el Covid-19 contribuyó para que en el primer trimestre del año el Producto Interno Bruto (PIB) cayera un 2.4% respecto al mismo trimestre del 2019 y la variación anual de los sectores secundario y terciario fluctuara entre los -3.4% y -1.8%.

Según el Censo Económico de 2019, en México predominan las microempresas. Este tipo de empresas se caracterizan por contar con un número muy reducido de empleados (entre 1 y 2) lo que puede significar una debilidad en términos de capacidad financiera. Esta debilidad hace que para este tipo de empresas sea prácticamente imposible sobrellevar contingencias como lo es la pandemia. La pobre capacidad financiera de las microempresas fue más que evidente ya que, miles de personas en México perdieron su trabajo o no recibieron salario durante meses.

En términos sociales, la crisis afectará a México incrementando la brecha en términos de aprendizaje. Cómo sabemos, México es un país en dónde la mayor parte de su población vive pobreza de alguna forma, lo que provocará que millones de personas vivan dificultades para adaptarse a las nuevas modalidades de trabajo y escuela no presenciales.

Por otro lado, según la ZMVM la violencia intrafamiliar aumentó un 7.2% y el número de reportes de violencia familiar registró un aumento del 24%. Estos números resultan alarmantes para todos los mexicanos por lo que es necesario tomar acciones que nos lleven a superar la pandemia lo más pronto posible.

Si bien la aplicación que vamos a desarrollar no contribuye directamente a la solución de los problemas señalados en los párrafos anteriores, si lo hará de forma indirecta. Cómo ya se dijo, la aplicación busca tener informados a los mexicanos, para así poder reducir el número de personas contagiadas y salir de la situación actual lo más pronto posible.

(Ríos, 2019)

El ITESO consciente de la importancia que tiene el bosque de la primavera en la vida de los ciudadanos de Guadalajara se ha preocupado por el estado del bosque, es por eso por lo que se han creado proyectos como la red de sensores del PAP vida digital. Con la WSN que ha ido creando el ITESO a través de los años se ha aprovechado las tecnologías para poder

monitorear el estado actual del bosque en ciertos puntos del ITESO. Alumnos de diferentes disciplinas como ingeniería en sistemas computacionales, ingeniería electrónica, ingeniería mecánica, diseño, entre otras, aportan sus conocimientos para que el proyecto sea más completo y tenga mayor impacto. Actualmente el proyecto cuenta con una red de 15 nodos repartidos entre el ITESO y el bosque, un sitio web para consultar los datos conseguidos por los sensores de forma organizada y entendible, y servicios web. La fase de desarrollo móvil aún no se ha comenzado, pero se cuenta con una idea de diseño que se va a revisar y perfeccionar para después iniciar con el desarrollo de la aplicación móvil.

2. Desarrollo

2.1. Sustento teórico y metodológico

POO

La programación orientada a objetos es un paradigma de programación que usa objetos en sus interacciones, para diseñar aplicaciones y programas informáticos. Un objeto consta de un estado y un comportamiento.Utiliza técnicas como herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.
(Vázquez, 2018)

Programación Funcional

La programación funcional se centra en las funciones. En un programa funcional, todos los elementos pueden entenderse como funciones y el código puede ejecutarse mediante llamadas de función secuenciales.
(1&1 IONOS Inc., 2020)

Rest API

Representational State Transfer (REST) es un estilo arquitectónico y una metodología utilizada para servicios web. Utiliza servicios HTTP para la comunicación entre el cliente y el servidor. La API rest implementa los métodos PUT, POST, GET, HEAD, DELETE, CONNECT, TRACE y PATCH.

(Naeem, 2020)

Desarrollo Móvil

El desarrollo de aplicaciones móviles son los procedimientos y procesos establecidos que intervienen cuando se crea software para pequeños dispositivos informáticos inalámbricos, como tabletas y teléfonos inteligentes.

(Feliciano, 2019)

Kotlin

Según *Android Developers*, Kotlin es lenguaje de programación *Open-Source* y estáticamente tipado que soporta dos de los paradigmas de programación; POO y programación funcional. Este lenguaje de programación ofrece sintaxis similar a otros lenguajes de programación cómo C#, Java y Scala.

Desde el 2019, Kotlin es oficialmente soportado por Google para Desarrollo Móvil Android, lo que significa que pasó a ser el lenguaje oficial para desarrollo nativo Android.

(Google, 2019)

XML

Extensible Markup Language es un metalenguaje que permite la organización y el etiquetado de documentos. Permite cambiar datos entre sistemas, guardar datos de forma estandarizada, Android Studio lo usa para crear las vistas de sus aplicaciones.

(Pérez & Gardey, 2010)

MVVM

Model View - View Model es una arquitectura de desarrollo creada por Microsoft en 2004. Busca el desacoplamiento entre la parte visual, la lógica del negocio y todo lo que lo relaciona. Facilita la realización de pruebas unitarias.

(Ortega, 2020)

Retrofit

Retrofit es un Cliente HTTP que facilita el manejo de errores en llamadas a servicios web (REST API) para Android y Java.

(Square Inc., 2013)

Scrum

Metodología Ágil en la que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

(ProyectosAgiles.org, 2018)

Jira Software

Jira Software es una herramienta utilizada por equipos ágiles de desarrollo de Software. Este software te permite cumplir con cuatro actividades básicas que se necesitan en todo proyecto de desarrollo. La primera de ellas es la Planificación, Jira Software te permite crear historias de usuarios, Sprints y distribuir tareas entre todo el equipo de software. Otra cosa que te permite JS es priorizar y a analizar el trabajo de los integrantes de todo el equipo mediante la creación de tableros. De igual forma, facilita la liberación del producto de software en sus distintas versiones y tener un proceso de control para esto. Finalmente, JS facilita la creación de reportes automáticos de forma sencilla.

(Atlassian, 2020)

GitHub

GitHub es un sistema de gestión de proyectos y control de versiones de código, así como una plataforma de red social diseñada para desarrolladores. GitHub utiliza Git, un sistema de control de versiones desarrollado por Linus Torvalds.

(B. 2019)

2.2. Planeación y seguimiento del proyecto

A. Descripción del proyecto

El proyecto que llevaremos a cabo consiste en el desarrollo de dos aplicaciones móviles. Para el desarrollo de las aplicaciones utilizaremos la metodología de trabajo ágil SCRUM y nos apoyaremos de Jira Software para la distribución de tareas y monitoreo del trabajo. El código fuente de las aplicaciones estará en repositorios de GitHub privados para que el control de versiones sea sencillo y podamos trabajar efectivamente por separado.

Las aplicaciones serán proyectos nativos de Android utilizando el lenguaje oficial soportado por Google Kotlin. Para comunicarnos con los servicios web desarrollados en verano 2020, utilizaremos el Cliente HTTP de Retrofit y lo integraremos con el patrón arquitectónico *Model View – View Model* (MVVM).

Las etapas en ambos proyectos son las siguientes:

I. Planeación
II. Desarrollo
III. Pruebas
IV. Publicación de Aplicación
V. Documentación (En paralelo con las etapas 1-4)
VI.

B. Plan de trabajo

Monitor Covid

Título tabla: Fechas de semanas Monitor Covid.

Descripción: Tabla donde se indica las fechas correspondientes a cada número de semana del plan de trabajo.

Semana	Inicio	Fin
1	2-Sep-20	8-Sep-20
2	9-Sep-20	15-Sep-20
3	16-Sep-20	22-Sep-20

4	23-Sep-20	29-Sep-20
5	30-Sep-20	6-Oct-20
6	7-Oct-20	13-Oct-20
7	14-Oct-20	20-Oct-20
8	21-Oct-20	27-Oct-20
9	28-Oct-20	3-Nov-20
10	4-Nov-20	10-Nov-20
11	11-Nov-20	17-Nov-20
12	18-Nov-20	24-Nov-20

Título tabla: Plan de trabajo Monitor Covid.

Descripción: Plan de trabajo de la aplicación donde se divide por productos y las actividades necesarias para llegar a ellos. Se especifican tiempos, responsables, tipos de tareas y fecha en la que se trabajará.

Producto	Actividades	RH	Tipo	Tiempo	Semana
Maquetado Aplicación	Crear proyecto	Isaac Cabrera	Desarrollo	0.25	1
	Configurar Build Variants	Isaac Cabrera	Desarrollo	1	1
	Importar XMLs	Isaac Cabrera	Desarrollo	2	1
	Crear Actividades para XML	Isaac Cabrera	Desarrollo	1.5	1
	Agregar paleta de colores	Isaac Cabrera	Desarrollo	0.5	1
	Crear tema de aplicación	Isaac Cabrera	Desarrollo	1	1
	Investigación MVVM	Isaac Cabrera, Alessandro Pallaro	Desarrollo	5	1
	Registrar aplicación en google console	Isaac Cabrera	Desarrollo	1	1
Pantalla Carga	Refactorizar XML	Alessandro Pallaro	Desarrollo	1	1
	Animación de carga	Alessandro Pallaro	Desarrollo	1	1
	View Model	Alessandro Pallaro	Desarrollo	3	9
	Flujo a siguiente actividad	Alessandro Pallaro	Desarrollo	0.25	1
Pantalla Presentación	Refactorizar XML	Alessandro Pallaro	Desarrollo	1	2
	Flujo a siguiente actividad	Alessandro Pallaro	Desarrollo	0.25	2
	Validar mostrar solo la primera vez	Alessandro Pallaro	Desarrollo	1	9

Pantalla Registro	Refactorizar XML	Isaac Cabrera	Desarrollo	1	2
	Validar registro previo	Isaac Cabrera	Desarrollo	1	9
	View Model	Isaac Cabrera	Desarrollo	3	9
	Registro de usuario	Isaac Cabrera	Desarrollo	2	9
	Flujo a siguiente actividad	Isaac Cabrera	Desarrollo	0.25	2
Pantalla Folio	Refactorizar XML	Isaac Cabrera	Desarrollo	1	2
	View Model	Isaac Cabrera	Desarrollo	3	10
	Inicio sesión	Isaac Cabrera	Desarrollo	2	10
	Flujo a siguiente actividad	Isaac Cabrera	Desarrollo	0.25	2
Pantalla Menu	Refactorizar XML	Alessandro Pallaro	Desarrollo	1	1
	Flujo a opciones del menu	Alessandro Pallaro	Desarrollo	1	1
Pantalla Mapa General	Refactorizar XML	Isaac Cabrera	Desarrollo	1	2
	Habilitar Google Maps API en la Google Console	Isaac Cabrera	Desarrollo	1	2
	Crear API Key para google maps	Isaac Cabrera	Desarrollo	0.5	2
	Integrar Map Fragment	Isaac Cabrera	Desarrollo	2	4
	View Model	Isaac Cabrera	Desarrollo	3	10
	Mostrar markers/overlays con información	Isaac Cabrera	Desarrollo	5	10
Pantalla Test	Main Fragment XML	Alessandro Pallaro	Desarrollo	4	3
	Fragmento preguntas XML	Alessandro Pallaro	Desarrollo	6	3
	Fragmento Resultados XML	Alessandro Pallaro	Desarrollo	6	6
	Fragmento volcer hacer test XML	Alessandro Pallaro	Desarrollo	6	6
	View Model	Alessandro Pallaro	Desarrollo	6	9
	Mandar test	Alessandro Pallaro	Desarrollo	6	9
	Mostrar resultados	Alessandro Pallaro	Desarrollo	4	10
	Navegación a otras actividades	Alessandro Pallaro	Desarrollo	0.25	4
Pantalla Novedades	Refactorizar XML	Alessandro Pallaro	Desarrollo	1	1
	Mostrar datos	Alessandro Pallaro	Desarrollo	4	10
	View Model	Alessandro Pallaro	Desarrollo	6	11
Pantalla Estadisticas	Refactorizar XML	Isaac Cabrera	Desarrollo	1	3

	Investigación Graficas en Android	Isaac Cabrera	Desarrollo	4	3
	Creación de graficas	Isaac Cabrera	Desarrollo	6	3
	View Model	Alessandro Pallaro	Desarrollo	3	11
	Navegación a otras actividades	Isaac Cabrera	Desarrollo	0.25	2
Pantalla Recomendaciones	Refactorizar XML	Alessandro Pallaro	Desarrollo	1	2
Pantalla Hospitales	Integrar Map Fragment	Isaac Cabrera	Desarrollo	2	5
	View Model	Isaac Cabrera	Desarrollo	2	11
	Mostrar markers/overlays con información	Isaac Cabrera	Desarrollo	5	11
Pantalla Acerca de	Refactorizar XML	Alessandro Pallaro	Desarrollo	1	2
Pantalla Ajustes	Refactorizar XML	Isaac Cabrera	Desarrollo	1	2
	Cerrar sesión	Isaac Cabrera	Desarrollo	1	11
Repositorios	Repositorio Auth	Isaac Cabrera	Desarrollo	4	8
	Repositorio Pruebas	Isaac Cabrera	Desarrollo	4	8
	Repo Novedades	Isaac Cabrera	Desarrollo	4	8
	Repositorio Mapas	Isaac Cabrera	Desarrollo	4	9
	Investigación Retrofit	Isaac Cabrera, Alessandro Pallaro	Desarrollo	4	2
Testing	Pruebas	Isaac Cabrera, Alessandro Pallaro	Desarrollo	6	12
Deployment	Investigación	Isaac Cabrera, Alessandro Pallaro	Desarrollo	2	12
	Sign APK	Isaac Cabrera, Alessandro Pallaro	Desarrollo	2	12
	Upload App to PS	Isaac Cabrera, Alessandro Pallaro	Desarrollo	4	12
Dialogos	Cargando	Isaac Cabrera	Desarrollo	2	4
	Error	Isaac Cabrera	Desarrollo	2	4
Arreglar warnings y estilo	App warnings	Isaac Cabrera, Alessandro Pallaro	Desarrollo	4	4
Spinners	Folio	Isaac Cabrera	Desarrollo	1	4
	Novedades	Alessandro Pallaro	Desarrollo	1	4
Revisión Backend	Revisar y documentar estado de los Servicios Web	Isaac Cabrera, Alessandro Pallaro	Desarrollo	10	5
Endpoint Usuario	Controller	Isaac Cabrera	Desarrollo	3	6
	Modelo	Isaac Cabrera	Desarrollo	5	6

	Router	Isaac Cabrera	Desarrollo	1	6
Endpoint Prueba	Controller	Isaac Cabrera	Desarrollo	3	7
	Modelo	Isaac Cabrera	Desarrollo	6	7
	Router	Isaac Cabrera	Desarrollo	1	7
Endpoint Datosgob	Controller	Alessandro Pallaro	Desarrollo	5	7
	Modelo	Alessandro Pallaro	Desarrollo	5	7
	Router	Alessandro Pallaro	Desarrollo	2	7
Endpoint Recomendaciones	Controller	Alessandro Pallaro	Desarrollo	2	8
	Modelo	Alessandro Pallaro	Desarrollo	2	8
	Router	Alessandro Pallaro	Desarrollo	2	8
Endpoint Mapa	Controller	Alessandro Pallaro	Desarrollo	2	8
	Modelo	Alessandro Pallaro	Desarrollo	2	8
	Router	Alessandro Pallaro	Desarrollo	2	8

WSN App

Título tabla: Fechas de semanas WSN App.

Descripción: Tabla donde se indica las fechas correspondientes a cada número de semana del plan de trabajo.

Semana	Inicio	Fin
1	25-Nov-20	1-Dec-20
2		
3		
4		
5		
6		
7		
8		

Título tabla: Plan de trabajo WSN App.

Descripción: Plan de trabajo de la aplicación donde se divide por productos y las actividades necesarias para llegar a ellos. Se especifican tiempos, responsables, tipos de tareas y fecha en la que se trabajará.

Producto	Actividades	RH	Tipo	Tiempo	Semana
Maquetado Aplicación	Crear proyecto	Isaac Cabrera	Técnica	0.25	1
	Configurar Build Variants	Isaac Cabrera	Técnica	1	1
	Importar XMLs	Isaac Cabrera	Técnica	2	1
	Crear Actividades para XML	Isaac Cabrera	Técnica	1.5	1
	Agregar paleta de colores	Isaac Cabrera	Técnica	0.5	1
	Crear tema de aplicación	Isaac Cabrera	Técnica	1	1
	Investigación MVVM	Isaac Cabrera, Alessandro Pallaro	Profesional	5	1
	Registrar aplicación en google console	Isaac Cabrera	Técnica	1	2
Pantalla de carga	XML	Alessandro Pallaro	Técnica	1	1
	Actividad	Alessandro Pallaro	Técnica	1	1
	Animación drone	Alessandro Pallaro	Técnica	3	1
Pantalla menú	Custom Navigation View	Alessandro Pallaro	Técnica	15	2
	Recycler Creencia, Tecnica	Isaac Cabrera	Técnica	6	2
	View Model	Alessandro Pallaro	Técnica	2	3
	Recycler Galeria	Isaac Cabrera	Técnica	6	2
Pantalla creencia	XML	Alessandro Pallaro	Técnica	1	3
	Actividad	Alessandro Pallaro	Técnica	1	3
Pantalla técnica	XML	Isaac Cabrera	Técnica	1	3
	Actividad	Isaac Cabrera	Técnica	1	3
Pantalla nosotros	XML	Alessandro Pallaro	Técnica	1	3
	Actividad	Alessandro Pallaro	Técnica	1	3
Pantalla Detalle Galería	XML	Isaac Cabrera	Técnica	1	3
	Actividad	Isaac Cabrera	Técnica	1	3
Mapa	XML	Isaac Cabrera	Técnica	2	3
	Habilitar Google Maps API en la Google Console	Isaac Cabrera	Técnica	1	3
	Crear API Key para google maps	Isaac Cabrera	Técnica	0.5	3
	Integrar Map Fragment	Isaac Cabrera	Técnica	2	3

	View Model	Isaac Cabrera	Técnica	2	3
	Mostrar markers/overlays con información	Isaac Cabrera	Técnica	10	4
	Floating Menu	Alessandro Pallaro	Técnica	15	4
Pantalla Detalle Nodo	XML	Isaac Cabrera	Técnica	2	4
	Filtros	Isaac Cabrera	Técnica	1	5
	Investigación Graficas	Isaac Cabrera	Profesional	6	5
	Graficas	Isaac Cabrera	Técnica	4	5
Contacto	XML	Isaac Cabrera	Técnica	2	6
	Actividad	Isaac Cabrera	Técnica	1	6
	Enviar correo	Isaac Cabrera	Técnica	1	6
Repositorio	Investigación Retrofit	Alessandro Pallaro	Profesional	10	5
	Implementación	Alessandro Pallaro	Técnica	10	6
Pruebas	Pruebas	Isaac Cabrera, Alessandro Pallaro	Técnica	12	7
Liberación	Investigación	Isaac Cabrera, Alessandro Pallaro	Profesional	6	8
	Sign APK	Isaac Cabrera, Alessandro Pallaro	Técnica	2	8
	Upload App to PS	Isaac Cabrera, Alessandro Pallaro	Técnica	4	8

Título tabla: Reuniones con el equipo.

Descripción: En esta tabla se lleva el registro de todas aquellas reuniones que se tuvieron con alumnos integrantes del PAP. Se registra la fecha de la reunión, su objetivo, duración y los participantes.

Fecha de la reunión	Objetivo	Duración	Participantes
Septiembre 3	Realizar plan de trabajo	2h36m	Isaac Cabrera, Alessandro Pallaro
Septiembre 6	Acordar como se trabajará en cuanto a GitHub, Jira y Android Studio	1h27m	Isaac Cabrera, Alessandro Pallaro
Septiembre 8	Presentar avances personales	25m	Isaac Cabrera, Alessandro Pallaro
Septiembre 8	Definir trabajo de la semana	17m	Isaac Cabrera, Alessandro Pallaro
Septiembre 10	Trabajar en el reporte PAP	1h15m	Isaac Cabrera, Alessandro Pallaro

Septiembre 11	Integrar mapa en proyecto de Android Studio	1h3m	Isaac Cabrera, Alessandro Pallaro
Septiembre 15	Presentar avances personales	28m	Isaac Cabrera, Alessandro Pallaro
Septiembre 15	Definir trabajo de la semana	15m	Isaac Cabrera, Alessandro Pallaro
Septiembre 17	Trabajar en el reporte PAP	47m	Isaac Cabrera, Alessandro Pallaro
Septiembre 22	Presentar avances personales	13m	Isaac Cabrera, Alessandro Pallaro
Septiembre 22	Definir trabajo de la semana	19m	Isaac Cabrera, Alessandro Pallaro
Septiembre 24	Trabajar en reporte PAP	23m	Isaac Cabrera, Alessandro Pallaro
Septiembre 24	Discutir trabajo con *Backend*	1h	Isaac Cabrera, Alessandro Pallaro, Carlos Soto, Héctor Chávez
Septiembre 28	Presentar avances personales	20m	Isaac Cabrera, Alessandro Pallaro
Septiembre 28	Discutir sobre fallas que presenta *Backend*	30m	Isaac Cabrera, Alessandro Pallaro
Octubre 1	Planeación para refactorizar el *Backend* para la aplicación Monitor Covid.	1h	Isaac Cabrera, Alessandro Pallaro, Carlos Soto
Octubre 5	Creación del plan de trabajo para refactorizar el *Backend* de la aplicación monitor Covid.	2h	Isaac Cabrera, Alessandro Pallaro, Carlos Soto
Octubre 13	Refactorizar esquema de la base datos. Definir la forma del folio.	1h 30m	Isaac Cabrera, Carlos Soto
Octubre 20	Definir la autenticación para el *Backend* utilizando JWT	1h	Isaac Cabrera, Carlos Soto, Héctor Chávez y Alessandro Pallaro
Octubre 22	Revisar avances del proyecto, redefinir el plan de trabajo.	1h	Isaac Cabrera, Alessandro Pallaro

Octubre 30	Definir el funcionamiento de los servicios web del mapa.	1h	Isaac Cabrera, Alessandro Pallaro
Noviembre 22	Integración YELP API en la aplicación	4h	Isaac Cabrera, Alessandro Pallaro

Título tabla: Reuniones con el profesor.

Descripción: En esta tabla se lleva el registro de todas aquellas reuniones que se tuvieron con el profesor del PAP. Se registra la fecha de la reunión, su objetivo, duración y los participantes.

Fecha de la reunión	Objetivo	Duración	Participantes
Septiembre 8	Sesión clase para presentar avances (maquetado proyecto)	4h	Todos
Septiembre 15	Sesión clase para presentar avances (refactorizado XML)	4h	Todos
Septiembre 22	Sesión clase para presentar avances (refactorizado XML)	4h	Todos
Septiembre 25	Discutir sobre oportunidades de mejora del *Backend* existente		Profesor Luis Eduardo Pérez Bernal, Isaac Cabrera Cortes
Septiembre 29	Sesión clase para presentar avances (Test y repositorios)	4h	Todos
Octubre 6	Presentamos correcciones en ciertos aspectos de la UI y el plan de trabajo para la refactorización del *Backend*. Se propusieron nuevos cambios a ser implementados en la UI.	1h	Todos
Octubre 13	Presentamos correcciones terminadas de la UI y avances de la implementación del *Backend*.	1h	Todos
Octubre 15	Revisión Reporte PAP	2h	Todos
Octubre 20	Presentamos avances sobre la refactorización del *Backend* para	2h	Todos

	aplicación Monitor Covid. Quedamos en iniciar los Repositorios de la Aplicación Móvil y terminar de documentar.		
Octubre 22	Revisar avances del reporte PAP y reflexión sobre los proyectos.	2h	Todos
Octubre 27	Presentar avances de la semana referentes a los servicios web y clientes http con Retrofit.	1h	Todos Ausente: Isaac Cabrera
Octubre 29	Revisar avances del reporte PAP y reflexión sobre los proyectos.	2h	Todos
Noviembre 3	Presentamos la integración de los servicios web para el inicio de sesión y registro terminada, así como el servicio web para el mapa. Compartimos el trabajo por hacer, servicio web para gráficas e integrar los otros repositorios a la aplicación.	1h	Todos
Noviembre 5	Revisar avances del reporte PAP.	1h	Todos
Noviembre 10	Presentar avances en la aplicación Pruebas y Estadísticas, cambios en Ajustes y problemas con Places API para mostrar los hospitales en el mapa.	1h	Todos
Noviembre 12	Revisión Reporte PAP	1h	Todos
Noviembre 17	Presentar avances, mapa general terminado y presentar alternativas para la parte de hospitales.	1h	Todos
Noviembre 19	Revisión Reporte PAP	1h	Todos
Noviembre 24	Presentar versión 1.0 de la aplicación terminada. Pendiente hacer liberación de la aplicación.	2h	Todos
Noviembre 26	Presentar avances de reporte PAP. Problemas con cuenta de Google.	2h	Todos

	Revisar propuesta de cartel del equipo de diseño		

C. Desarrollo de propuesta de mejora

Monitor Covid

I. Maquetado Aplicación

Se muestra el *Gradle* de la aplicación, archivo en dónde se especifican las dependencias de la aplicación y principales configuraciones.

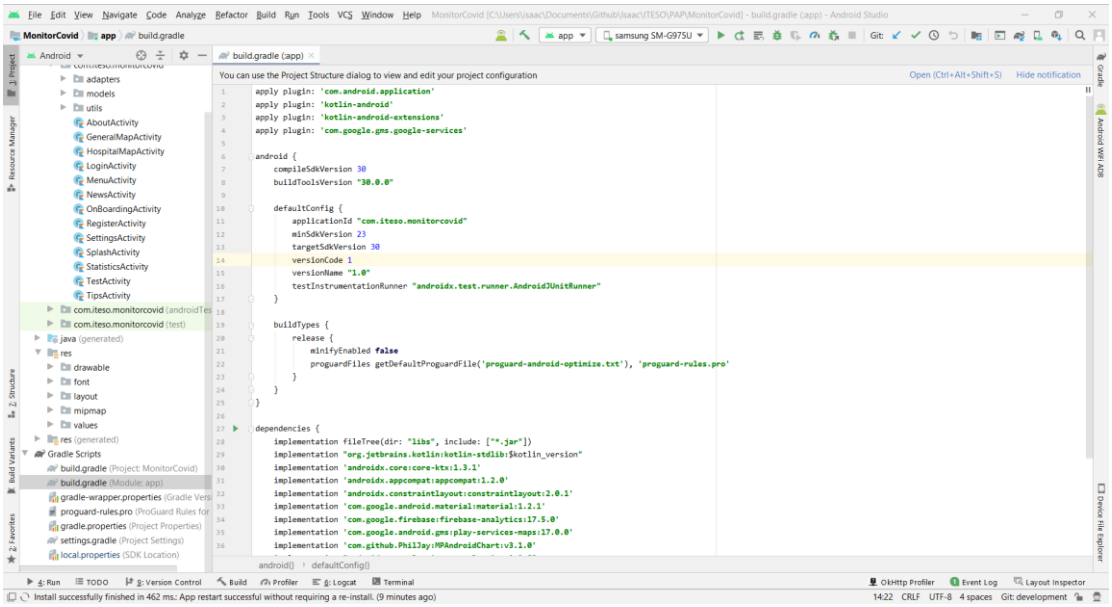


Ilustración 1 Maquetado de la aplicación

II. Pantalla Carga

Esta pantalla muestra por unos segundos para esperar a que cargue la aplicación. Se usó un *RelativeLayout* para añadir dos *ImageView* que generen la vista en el XML, y en el Kotlin se agregó un *Handler* para mostrarlo solo unos segundos y se cambie automáticamente.

Ilustración 2 Pantalla de inicio

III. Pantalla Presentación

La pantalla de presentación muestra al usuario las funcionalidades principales de la aplicación. Esta pantalla solo se muestra la primera vez que el usuario entre a la aplicación después de descargarla. Para lograr esta vista se utilizó un *ConstraintLayout* en el que se le añadieron los iconos de cada función como *ImageView*, y sus respectivas descripciones como *TextView*, también cuenta con dos botones, el primero registrarte y el segundo para iniciar sesión.

Ilustración 3 Pantalla de presentación

IV. Pantalla Registro

Pantalla de la aplicación en la que el usuario se puede registrar y solicitar un folio para poder identificarse de forma anónima. Está pantalla utiliza un *ConstraintLayout* como padre y vistas cómo *TextView*, *EditText*, *Button, ImageButton y CheckBox.*

En la imagen se muestra el cambio en el campo Sexo. Anteriormente el campo era un campo de texto y se hizo el cambio por un *Spinner* para evitar que el usuario indique un valor inválido.

Ilustración 4 Registro v1.0

Ilustración 5 Registro v2.0

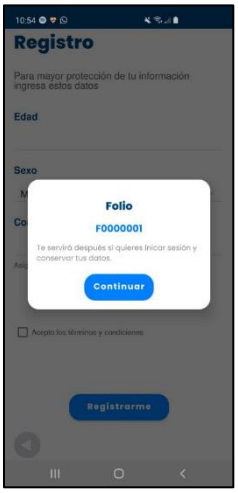


Ilustración 6 Resultado de registro

V. Pantalla Folio

Pantalla en la que el usuario puede iniciar sesión de forma anónima en la aplicación. Esta pantalla utiliza un *ConstraintLayout* como contenedor padre y vistas cómo *TextView, EditText, Button, ImageButton y CheckBox.*

Ilustración 7 Pantalla de inicio de sesión

VI. Pantalla Menú

En esta pantalla el usuario puede acceder a todas las funcionalidades de la aplicación. Para la realización de esta pantalla se utilizó un *ConstraintLayout* general que contiene un *ConstraintLayout* por cada imagen con botón que hay. Se utilizaron *Shapes* para darle el efecto de esquina redondeada a los botones. En el Kotlin se asignó el flujo de cada botón a su actividad correspondiente.

Ilustración 8 Pantalla menú principal

VII. Pantalla Mapa General

Pantalla en la que el usuario podrá áreas con alta probabilidad de contagio. Estas áreas se irán generando, dependiendo de las pruebas realizadas en la aplicación. En está pantalla se utiliza un *ConstraintLayout* como contenedor padre y vistas cómo *Button*, *ImageButton*, *FloatingButton*, *TextView*. Para mostrar se utiliza el *SupportFragmentMap* del SDK de GoogleMaps.

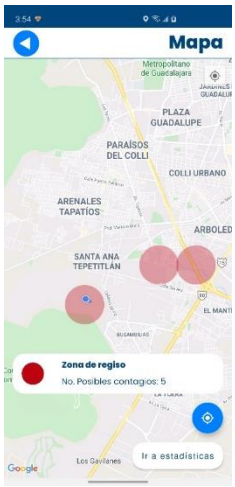


Ilustración 9 Mapa de zonas de riesgo

VIII. Pantalla Test

Las pantallas que se utilizaron para la prueba se realizaron con Fragmentos. Estas pantallas son la de introducción a la prueba, la prueba, los resultados de la prueba y repetir la prueba.

Fragmento Introducción test

Esta pantalla se realizó como fragmento, en ella se muestra alguna información sobre la prueba, un botón para iniciar la prueba y otro que te redireccionará a hospitales.

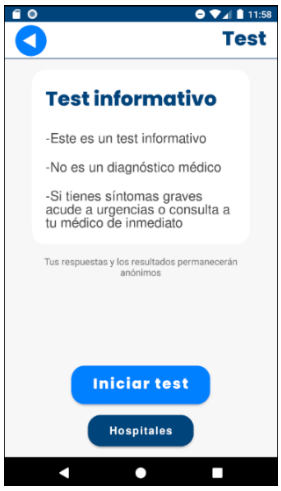


Ilustración 10 Fragmento para introducción de pruebas

IX. Fragmento test

El fragmento test se realizó para poder mostrar cualquier prueba que se envíe desde *Backend*. El número de preguntas y de opciones que tendrá cada pregunta es dinámico. En cuanto a la vista se utilizó un *ConstraintLayout* y dentro se usó un RecyclerView para mostrar las opciones de las preguntas, para esto fue necesario crea el *Adapter* y la clase de Opciones, se programó la lógica de los botones para cambiar de pregunta cada que se pica siguiente o anterior, esconderse y mostrarse cuando sea necesario, y terminar la prueba y pasar a los resultados.

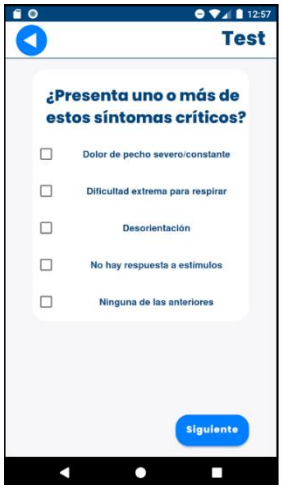


Ilustración 11 Fragmento de pruebas

X. Fragmento resultados

Al terminar la prueba se enviarán las respuestas al *Backend* y se obtendrán los resultados. Esta pantalla es capaz de mostrar cualquier resultado que se le envíe. Se utilizaron *TextView* para los títulos, y un *RecyclerView* para las recomendaciones.

Ilustración 12 Fragmento de resultado de prueba

XI. Fragmento Repetir

La pantalla repetir muestra los resultados de la última prueba realizado y le da la opción al usuario de repetir la prueba. Por medio de *SharedPreferences* se detecta si el usuario ya hizo una prueba anteriormente y cuál fue su resultado. La actividad está construida en su XML por un *ConstraintLayout* el cual contiene *TextView* para los títulos y un *RecyclerView* con las recomendaciones correspondientes del resultado.

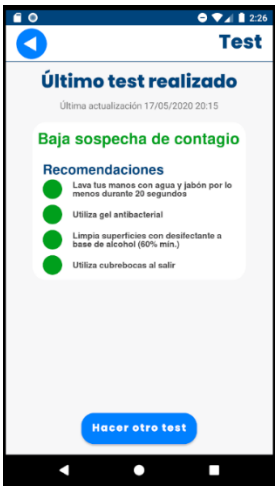


Ilustración 13 Fragmento para repetir la prueba

XII. Pantalla Novedades

La pantalla de novedades muestra las estadísticas sobre los casos confirmados, defunciones, casos descartados y casos sospechosos que se obtengan desde el *Backend*, también se tienen dos botones para consultar las estadísticas en otras fuentes de confianza. También se cuenta con un *Spinner* para seleccionar el estado del que se quiera revisar los datos.

Ilustración 14 Pantalla de novedades

XIII. Pantalla Estadísticas

Pantalla en la que se muestran estadísticas de las pruebas realizadas. Para la creación de gráficas se hace uso de una Librería *Open Source* llamada *MPAndroidChart*. El contendor padre es un *ConstraintLayout* y se agregan las gráficas de tipo *LineChart* y *PieChart*.

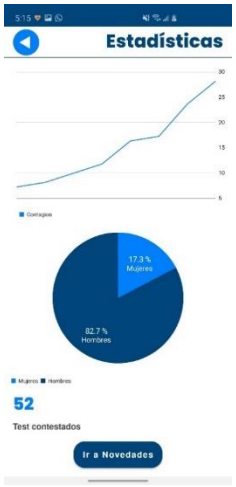


Ilustración 15 Pantalla de estadísticas

XIV. Pantalla Recomendaciones

Pantalla en las que se muestran las recomendaciones generales. En la Pantalla se utiliza un *ConstraintLayout* como contenedor padre y la lista de Tips se hizo utilizando un RecyclerView, lo que implicó crear los *ViewHolder*, *Adapter* y datos.

Ilustración 16 Pantalla de recomendaciones

XV. Pantalla Hospitales

Pantalla en la que el usuario podrá consultar los hospitales cercanos a su ubicación actual. En está pantalla se utiliza un *ConstraintLayout* como padre y vistas cómo *Button*, *ImageButton*, *FloatingButton*, *TextView*. Para mostrar se utiliza el *SupportFragmentMap* del SDK de GoogleMaps.

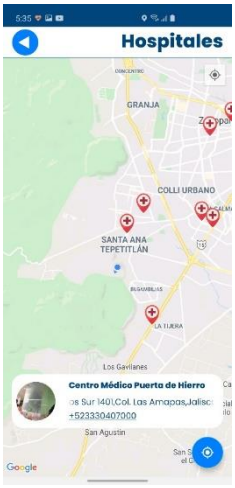


Ilustración 17 Pantalla de hospitales

XVI. Pantalla Acerca de

En la pantalla de acerca de se muestra la información sobre el proyecto, y unos botones para revisar el aviso de privacidad y los términos y condiciones.

Ilustración 18 Acerca de v1.0

Ilustración 19 Acerca de v2.0

Ilustración 20 Aviso de privacidad

XVII. Pantalla Ajustes

En pantalla de ajustes se pueden ajustar los radios de las áreas de peligro, para esto se utiliza un Slider con 4 pasos. El usuario también puede activar/desactivar los permisos de la ubicación dando clic al *Switch*. Finalmente, desde está pantalla se puede cerrar sesión dando clic a botón creado con una vista de tipo *Button*.

Ilustración 21 Pantalla ajustes v1.0

Ilustración 22 Pantalla ajustes v2.0

Ilustración 23 Pantalla ajustes v3.0

XVIII. Diálogo de Error

Para este diálogo de error se utilizó un *ConstraintLayout* como contenedor padre. Los elementos del diálogo son un *ImageView* para el ícono de error, dos *TextView* para el título y mensaje, y un botón para poder cerrarlo. Este diálogo de tipo *FragmentDialog* ya que nos permite tener mayor personalización de la UI.

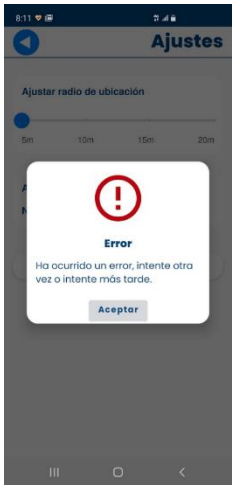


Ilustración 25 Diálogo de error v1.0

Ilustración 24 Diálogo de error v2.0

XIX. Diálogo de Carga

Para este diálogo de error se utilizó un *ConstraintLayout* como contenedor padre. Los elementos del diálogo son un *ProgressView* para indicar la carga y un *TextView* para el mensaje. Este diálogo de tipo *FragmentDialog* ya que nos permite tener mayor personalización de la UI.

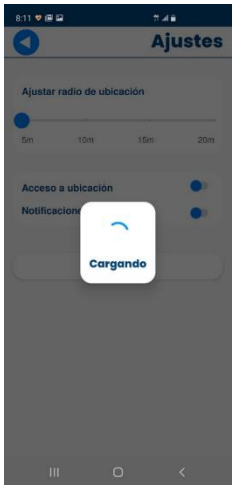
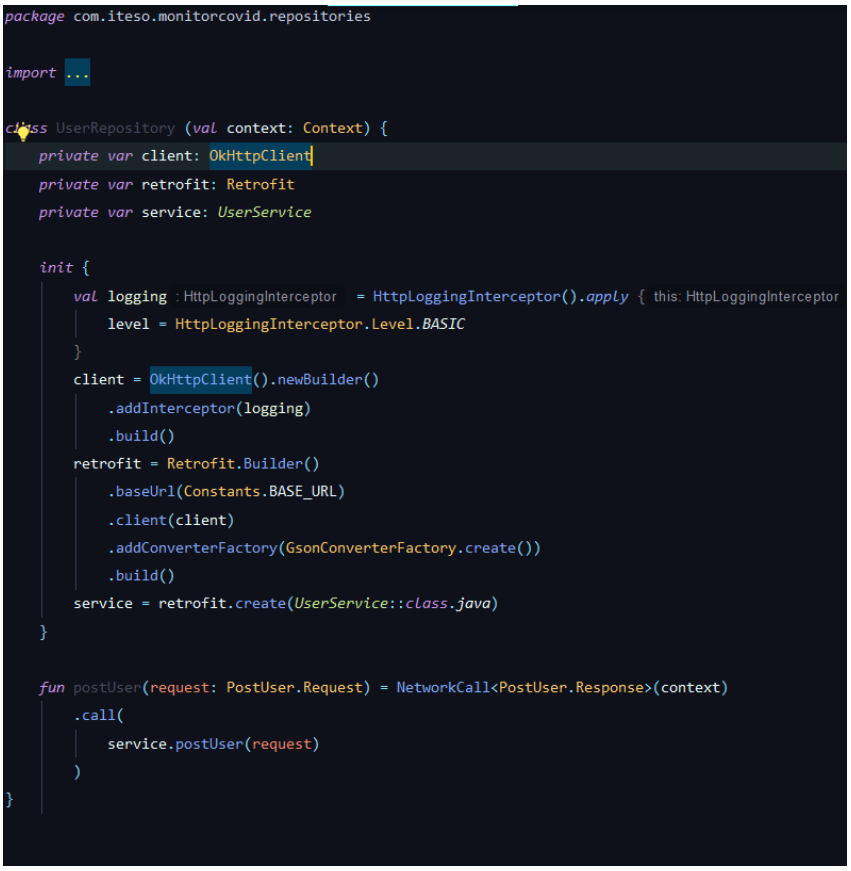


Ilustración 26 Diálogo de carga

XX. Repositorios

a. Repositorio de Usuarios

El repositorio de pruebas es el cliente http de Retrofit creado para interactuar con los servicios web del usuario. Para poder utilizar el Cliente fue necesario definir lo modelos del *Request* y *Response* del Servicio y crear una *Interfaz* para describir todos los Servicios que vamos a utilizar. Este repositorio se va a utilizar en la pantalla de registro de usuario.



```
package com.iteso.monitorcovid.repositories

import ...

class UserRepository (val context: Context) {
    private var client: OkHttpClient
    private var retrofit: Retrofit
    private var service: UserService

    init {
        val logging : HttpLoggingInterceptor  = HttpLoggingInterceptor().apply { this: HttpLoggingInterceptor
            level = HttpLoggingInterceptor.Level.BASIC
        }
        client = OkHttpClient().newBuilder()
            .addInterceptor(logging)
            .build()
        retrofit = Retrofit.Builder()
            .baseUrl(Constants.BASE_URL)
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        service = retrofit.create(UserService::class.java)
    }

    fun postUser(request: PostUser.Request) = NetworkCall<PostUser.Response>(context)
        .call(
            service.postUser(request)
        )
}
```

Ilustración 27 Repositorio de usuarios

b. Repositorio de Pruebas

El repositorio de pruebas es el cliente http de Retrofit creado para interactuar con los servicios web de las pruebas o *test*. Para poder utilizar el Cliente fue necesario definir lo modelos del *Request* y *Response* del Servicio y crear una *Interfaz* para describir todos los Servicios que vamos a utilizar. Este repositorio se va a utilizar en la pantalla de hacer *Test* o Prueba.

```
package com.iteso.monitorcovid.repositories

import ...

class PruebasRepository(val context: Context) {
    private var client: OkHttpClient
    private var retrofit: Retrofit
    private var service: PruebasService

    init {
        val logging : HttpLoggingInterceptor  = HttpLoggingInterceptor().apply { this: HttpLoggingInterceptor
            level = HttpLoggingInterceptor.Level.BASIC
        }
        client = OkHttpClient().newBuilder()
            .addInterceptor(logging)
            .build()
        retrofit = Retrofit.Builder()
            .baseUrl(Constants.BASE_URL)
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        service = retrofit.create(PruebasService::class.java)
    }

    fun postPrueba(request: PostPrueba.Request) = NetworkCall<PostPrueba.Response>(context)
        .call(
            service.postPrueba(request)
        )
}
```

Ilustración 28 Repositorio de pruebas

c. Repositorios de Autenticación

El repositorio de pruebas es el cliente http de Retrofit creado para interactuar con los servicios de autenticación de usuario. Para poder utilizar el Cliente fue necesario definir lo modelos del *Request* y *Response* del Servicio y crear una *Interfaz* para describir todos los Servicios que vamos a utilizar. Este repositorio se va a utilizar en la pantalla de inicio de sesión para poder autenticar al usuario con su Folio y Contraseña.

```
package com.iteso.monitorcovid.repositories


import ...

class AuthRepository (val context: Context) {
    private var client: OkHttpClient
    private var retrofit: Retrofit
    private var service: AuthService

    init {
        val logging : HttpLoggingInterceptor  = HttpLoggingInterceptor().apply { this: HttpLoggingInterceptor
            level = HttpLoggingInterceptor.Level.BASIC
        }
        client = OkHttpClient().newBuilder()
            .addInterceptor(logging)
            .build()
        retrofit = Retrofit.Builder()
            .baseUrl(Constants.BASE_URL)
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        service = retrofit.create(AuthService::class.java)
    }

    fun login(request: AuthLogin.Request) = NetworkCall<AuthLogin.Response>(context)
        .call(
            service.login(request)
        )
}
```

Ilustración 29 Repositorio autenticación

d. Repositorio de Datos de Gobierno

El repositorio de pruebas es el cliente http de Retrofit creado para interactuar con los servicios web relaciona con los datos del Gobierno para la sección de novedades. Para poder utilizar el Cliente fue necesario definir lo modelos del *Request* y *Response* del Servicio y crear una *Interfaz* para describir todos los Servicios que vamos a utilizar. Este repositorio se va a utilizar en la pantalla de novedades.

```
package com.iteso.monitorcovid.repositories

import ...

class GovernmentDataRepository(val context: Context) {
    private var client: OkHttpClient
    private var retrofit: Retrofit
    private var service: GovernmentDataService

    init {
        val logging : HttpLoggingInterceptor  = HttpLoggingInterceptor().apply { this: HttpLoggingInterceptor
            level = HttpLoggingInterceptor.Level.BASIC
        }
        client = OkHttpClient().newBuilder()
            .addInterceptor(logging)
            .build()
        retrofit = Retrofit.Builder()
            .baseUrl(Constants.BASE_URL)
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        service = retrofit.create(GovernmentDataService::class.java)
    }

    fun getLatestDataByEntity(state: String) = NetworkCall<List<GovernmentData>>(context)
        .call(
            service.getLatestDataByEntity(state)
        )
}
```

Ilustración 30 Repositorio datos del gobierno

e. Repositorio de Comunicaciones

El repositorio de pruebas es el cliente http de Retrofit creado para interactuar con los servicios web las comunicaciones (números de emergencia cómo línea Covid). Para poder utilizar el Cliente fue necesario definir lo modelos del *Request* y *Response* del Servicio y crear una *Interfaz* para describir todos los Servicios que vamos a utilizar. Este repositorio se va a utilizar en la pantalla de mapa de hospitales.

```
package com.iteso.monitorcovid.repositories

import ...

class CommunicationRepository(val context: Context) {
    private var client: OkHttpClient
    private var retrofit: Retrofit
    private var service: CommunicationService

    init {
        val logging : HttpLoggingInterceptor = HttpLoggingInterceptor().apply { this: HttpLoggingInterceptor
            level = HttpLoggingInterceptor.Level.BASIC
        }
        client = OkHttpClient().newBuilder()
            .addInterceptor(logging)
            .build()
        retrofit = Retrofit.Builder()
            .baseUrl(Constants.BASE_URL)
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        service = retrofit.create(CommunicationService::class.java)
    }

    fun getAllCommunications() = NetworkCall<List<Communication>>(context)
        .call(
            service.getAllCommunications()
        )
}
```

Ilustración 31 Repositorio de comunicaciones

f. Repositorio de Recomendaciones

El repositorio de recomendaciones es el cliente http de Retrofit creado para interactuar con los servicios web de las recomendaciones. Para poder utilizar el Cliente fue necesario definir lo modelos del *Request* y *Response* del Servicio y crear una *Interfaz* para describir todos los Servicios que vamos a utilizar. Este repositorio se va a utilizar en la pantalla de pruebas / test para obtener las recomendaciones cuando el usuario manda una prueba y obtiene su escrutinio.

```
package com.iteso.monitorcovid.repositories

import ...

class RecommendationRepository(val context: Context) {
    private var client: OkHttpClient
    private var retrofit: Retrofit
    private var service: RecommendationService

    init {
        val logging : HttpLoggingInterceptor = HttpLoggingInterceptor().apply { this: HttpLoggingInterceptor
            level = HttpLoggingInterceptor.Level.BASIC
        }
        client = OkHttpClient().newBuilder()
            .addInterceptor(logging)
            .build()
        retrofit = Retrofit.Builder()
            .baseUrl(Constants.BASE_URL)
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        service = retrofit.create(RecommendationService::class.java)
    }

    fun getRecommendationsByLevel(level: Int) = NetworkCall<List<Recommendation>>(context)
        .call(
            service.getRecommendationsByLevel(level)
        )
}
```

Ilustración 32 Repositorio recomendaciones

g. Repositorio del mapa general

El repositorio de mapa general es el cliente http de Retrofit creado para interactuar con los servicios web de los clústeres a ser dibujados en el mapa general de la aplicación. Para poder utilizar el Cliente fue necesario definir lo modelos del *Response* del Servicio y crear una *Interfaz* para describir todos los Servicios que vamos a utilizar. Este repositorio se va a utilizar en la pantalla de mapa para mostrar las zonas de riesgo cercanas al usuario.

```
package com.iteso.monitorcovid.repositories

import android.content.Context
import com.iteso.monitorcovid.models.map.general.Cluster
import com.iteso.monitorcovid.services.GeneralMapService
import com.iteso.monitorcovid.utils.Constants
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class GeneralMapRepository(val context: Context) {
    private var client: OkHttpClient
    private var retrofit: Retrofit
    private var service: GeneralMapService

    init {
        val logging : HttpLoggingInterceptor = HttpLoggingInterceptor().apply { this: HttpLoggingInterceptor
            level = HttpLoggingInterceptor.Level.BASIC
        }
        client = OkHttpClient().newBuilder()
            .addInterceptor(logging)
            .build()
        retrofit = Retrofit.Builder()
            .baseUrl(Constants.BASE_URL)
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        service = retrofit.create(GeneralMapService::class.java)
    }

    fun getClusters(radius: String) = NetworkCall<List<Cluster>>(context)
        .call(
            service.getClusters(radius)
        )
}
```

Ilustración 33 Repositorio mapa de zonas de riesgo

h. Repositorio del mapa de hospitales

El repositorio de mapa general es el cliente http de Retrofit creado para interactuar con los servicios web que ofrece YELP API para mostrar los hospitales cercanos al usuario. Para poder utilizar el Cliente fue necesario definir lo modelos del *Response* del Servicio y crear una *Interfaz* para describir todos los Servicios que vamos a utilizar. Este repositorio se va a utilizar en la pantalla en la que mostramos hospitales cercanos a la ubicación del usuario.

```
import com.iteso.monitorcovid.services.BusinessSearchService
import com.iteso.monitorcovid.utils.Constants
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class BusinessSearchRepository (val context: Context) {
    private var client: OkHttpClient
    private var retrofit: Retrofit
    private var service: BusinessSearchService

    init {
        val logging : HttpLoggingInterceptor  = HttpLoggingInterceptor().apply { this: HttpLoggingInterceptor
            level = HttpLoggingInterceptor.Level.BASIC
        }
        client = OkHttpClient().newBuilder()
            .addInterceptor(logging)
            .build()
        retrofit = Retrofit.Builder()
            .baseUrl(Constants.BASE_URL_YELP)
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        service = retrofit.create(BusinessSearchService::class.java)
    }

    fun getBusinessSearch(auth: String, latitude: Double, longitude: Double, radius: Int, categories: String)  = NetworkCall<BusinessSearch.Response>(context)
        .call(
            service.getBusinessSearch(
                auth: "Bearer $auth",
                latitude,
                longitude,
                radius,
                categories
            )
        )
}
```

Ilustración 34 Repositorio hospitales

XXI. Servicios Web para la App Monitor Covid

a. Usuarios

Controlador de Usuarios

El en archivo controlador de usuarios obtenemos a conexión a la base de datos mediante el contexto de ejecución y se lo pasamos al modelo. En el controlador, manejamos los datos obtenidos mediante el modelo o los errores que hubo en la petición o con la consulta sobre la base de datos.

Funciones de controlador

- Obtener todos los usuarios
- Crear un usuario
- Obtener un usuario por su folio

Ilustración 35 Controlador de usuarios

Modelo de Usuarios

El en archivo modelo de usuarios llevamos a cabo la conexión a la base de datos y obtenemos, insertamos o eliminamos la información relacionada a uno o más usuarios.

Funciones del modelo

- Obtener todos los usuarios
- Crear un usuario
- Obtener un usuario por su folio

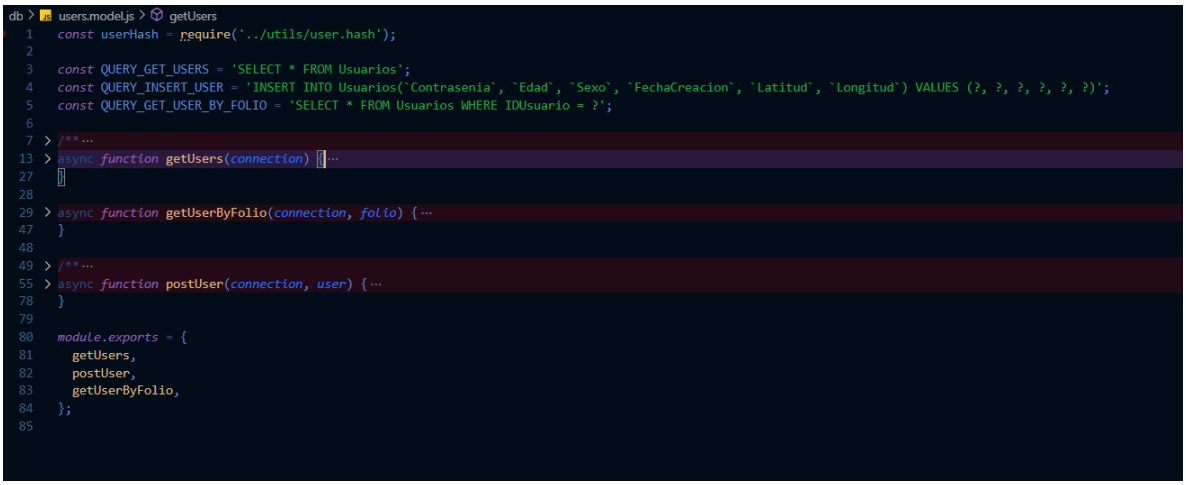


Ilustración 36 Modelo de usuarios

Router de usuarios

En este archivo definimos las rutas del *Backend* relacionadas con los usuarios. Manejamos las peticiones a ciertas rutas utilizando el controlador de usuarios.

Rutas

- Obtener todos los usuarios
- Crear un usuario
- Obtener un usuario por su folio

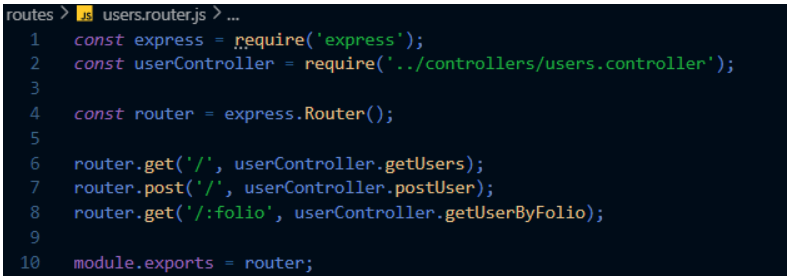


Ilustración 37 Router de usuarios

b. Pruebas

Controlador de pruebas

El en archivo controlador de pruebas obtenemos a conexión a la base de datos mediante el contexto de ejecución y se lo pasamos al modelo. En el controlador, manejamos los datos obtenidos mediante el modelo o los errores que hubo en la petición o con la consulta sobre la base de datos.

Funciones del controlador

- Obtener todas las pruebas
- Obtener una prueba por su ID
- Crear una prueba
- Eliminar una prueba por su ID
- Obtener todas las pruebas de un usuario por su folio
- Eliminar todas las pruebas de un usuario por su folio
- Actualizar una prueba por su ID

Ilustración 38 Controlador de pruebas

Modelo de pruebas

El en archivo modelo de pruebas llevamos a cabo la conexión a la base de datos y obtenemos, insertamos o eliminamos la información relacionada a uno o más pruebas.

Funciones del modelo

- Obtener todas las pruebas
- Obtener una prueba por su ID
- Crear una prueba
- Eliminar una prueba por su ID
- Obtener todas las pruebas de un usuario por su folio
- Eliminar todas las pruebas de un usuario por su folio

- Actualizar una prueba por su ID

Ilustración 39 Modelo de pruebas

Router de pruebas

En este archivo definimos las rutas del *Backend* relacionadas con las pruebas. Manejamos las peticiones a ciertas rutas utilizando el controlador de pruebas.

Rutas

- Obtener todas las pruebas
- Obtener una prueba por su ID
- Crear una prueba
- Eliminar una prueba por su ID

- Obtener todas las pruebas de un usuario por su folio
- Eliminar todas las pruebas de un usuario por su folio
- Actualizar una prueba por su ID

Ilustración 40 Router de pruebas

c. Validadores

Usuarios

Validador de modelos para usuarios. Todas las creaciones de usuarios son validadas para asegurar que los datos que mandas son válidos y cumplen con el esquema definido. En caso de que el modelo enviado por el cliente no es válido la operación en la base de datos no será realizado y se regresará el error.

Folio

Valida que todos los folios enviados en todas las peticiones sean válidos, es decir, que coincidan por el patrón definido. Esto se valida utilizando una expresión regular.

Prueba

Validador de modelos para pruebas. Todas las creaciones o modificaciones de pruebas son validadas para asegurar que los datos que mandas son

válidos y cumplen con el esquema definido. En caso de que el modelo enviado por el cliente no es válido la operación en la base de datos no será realizado y se regresará el error.

Util

Calculadora de Escrutinios

Módulo que sirve para calcular de forma eficiente el escrutinio de todas las pruebas que se mandan por los usuarios.

utils > escrutinio.calculator.js > compute

```
const WEIGHTS = [
  {
    weight: 100,
    fields: [
      'dolorSeveroPecho', 'difExtremaRespirar', 'desorientado', 'respEstimulos',
    ],
  },
  {
    weight: 10,
    fields: [
      'olfato', 'gusto', 'fiebre', 'escalofrios', 'respiracion', 'diarrea', 'vomito',
      'tos', 'dolorMuscular', 'dolorCabeza', 'irritacionOjos', 'sangradoRespiratorio',
    ],
  },
  {
    weight: 1,
    fields: [
      'Embarazada', 'consumeTabaco', 'enfCardiovascular', 'diabetes', 'cancer', 'obeso',
    ],
  },
];

function compute(prueba) {
  const keys = Object.keys(prueba);
  let escrutinio = 0;
  keys.forEach((key) => {
    WEIGHTS.forEach((item) => {
      if (item.fields.includes(key)) {
        if (prueba[key] > 0) {
          escrutinio += (prueba[key] + item.weight);
        }
      }
    });
  });
  if (escrutinio < 1) {
    return 1;
  }
  if (escrutinio <= 10) {
    return 2;
  }
  if (escrutinio <= 100) {
    return 3;
  }
  return 4;
}
```

Ilustración 41 Calculador de escrutinios

Generador de consultas

Módulo que sirve para crear consultas SQL de tipo *INSERT, UPDATE* de forma automática. Esto es útil para generar consultas sobre tablas como muchos campos de una forma sencilla.

utils > query.generator.js > <unknown>

```
function generateColumns(columns) {
  return `(${columns.join(',')})`;
}

function generatePlaceholders(count) {
  // eslint-disable-next-line prefer-spread
  const values = Array.apply(null, { length: count }).map(Function.call, () => '?').join(',');
  return `(${values})`;
}

function generateKeypairPlaceholders(columns) {
  if (columns.length === 1) {
    return columns.map((key) => `${key} = ?`);
  }
  return columns.map((key) => `${key} = ?`).join(',');
}

function generateInsertQuery(table, data) {
  const keys = Object.keys(data);
  const columns = generateColumns(keys);
  const values = generatePlaceholders(keys.length);
  const items = [];
  keys.forEach((key) => {
    items.push(data[key]);
  });
  return { query: `INSERT INTO ${table}${columns} VALUES ${values}`, values: items };
}

function generateUpdateQuery(table, setData, whereData) {
  const setKeys = Object.keys(setData);
  const setPlaceholders = generateKeypairPlaceholders(setKeys);
  const whereKeys = Object.keys(whereData);
  const wherePlaceHolders = generateKeypairPlaceholders(whereKeys);
  const values = [];
  setKeys.forEach((key) => {
    values.push(setData[key]);
  });
  whereKeys.forEach((key) => {
    values.push(whereData[key]);
  });
  return { query: `UPDATE ${table} SET ${setPlaceholders}  WHERE ${wherePlaceHolders}`, values };
}

```

Ilustración 42 Generador de consultas SQL

Mapeo de usuarios

Este módulo sirve para mapear el Id de un usuario a su folio y viceversa. Contiene dos funciones que hacen los mapeos respectivos.

Mapeo

Número => F(X) => Cadena 10 => F0000000A

Mapeo Inverso

Cadena => F-1(X) => Número F0000000A => 10

utils > user.hash.js > ...

```
function getFolioFromId(id) {
  const idHex = id.toString(16);
  let value = 'F';
  if (idHex.length < 8) {
    const zeros = 8 - idHex.length + 1;
    value += (Array(zeros).join('0') + idHex);
  }
  return value.toUpperCase();
}

function getIdFromFolio(folio) {
  const numericValue = folio.substring(1, folio.length);
  return parseInt(numericValue, 16);
}

module.exports = {
  getFolioFromId,
  getIdFromFolio,
};
```

Ilustración 43 Función Hash para mapeo de usuarios

d. Datos de gobierno

Controlador de Datosgov

El en archivo controlador de datosgov obtenemos la conexión a la base de datos mediante el contexto de ejecución y se lo pasamos al modelo. En el controlador, manejamos los datos obtenidos mediante el modelo o los errores que hubo en la petición o con la consulta sobre la base de datos.

Funciones de controlador

- Obtener todos los registros.
- Obtener el registro más reciente de cada estado.
- Crear un nuevo registro.
- Obtener el registro más reciente de un estado.
- Obtener todos los registros de un estado.
- Obtener todos los registros de una fecha específica.
- Obtener todos los registros de un estado en una fecha específica.

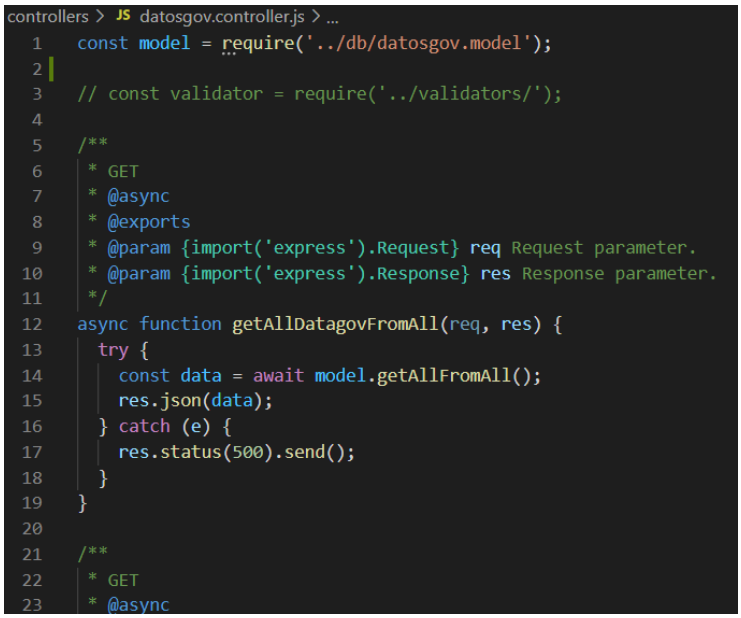


Ilustración 44 Controlador Datos del gobierno

Modelo de datosgov

En el archivo modelo de datosgov llevamos a cabo la conexión a la base de datos y obtenemos datos por estado o por fecha e insertamos datos nuevos.

Funciones del modelo

- Obtener todos los registros.
- Obtener el registro más reciente de cada estado.
- Crear un nuevo registro.
- Obtener el registro más reciente de un estado.
- Obtener todos los registros de un estado.
- Obtener todos los registros de una fecha específica.
- Obtener todos los registros de un estado en una fecha específica.

```
db > JS datosgov.model.js > getFromAllBySpecificDate
1  const { getConnection } = require('../config/dbConfig');
2
3  async function getAllFromAll() {
4    const connection = await getConnection();
5    return new Promise((resolve, reject) => {
6      connection.query('SELECT * FROM datosgob', (err, results) => {
7        if (err) {
8          reject(err);
9          return;
10       }
11       resolve(results);
12     });
13   });
14 }
15
16 async function getLatestFromAll() {
17   const connection = await getConnection();
18   return new Promise((resolve, reject) => {
19     connection.query('SELECT * FROM datosgob WHERE fecha IN (SELECT MAX(fecha) FROM datosgob GROUP BY Estado) ORDER BY Est
20       if (err) {
21         reject(err);
22         return;
23       }
```

Ilustración 45 Modelo Datos del gobierno

Rutas

- Obtener todos los registros.
- Obtener el registro más reciente de cada estado.
- Crear un nuevo registro.
- Obtener el registro más reciente de un estado.
- Obtener todos los registros de un estado.
- Obtener todos los registros de una fecha específica.
- Obtener todos los registros de un estado en una fecha específica.

```
routes > JS datosgov.router.js > ...
1  const express = require('express');
2
3  const controller = require('../controllers/datosgov.controller');
4
5  const router = express.Router();
6
7  router.get('/', controller.getAllDatagovFromAll);
8  router.get('/todos', controller.getLatestDatagovFromAll);
9  router.post('/', controller.postNewDatagov);
10 router.get('/entidad/:entidad', controller.getLatestDatagovByEntity);
11 router.get('/entidad/:entidad/todos', controller.getAllDatagovByEntity);
12 router.get('/fecha/:anio/:mes/:dia', controller.getDatagovFromAllBySpecificDate);
13 router.get('/entidad/:entidad/fecha/:anio/:mes/:dia', controller.getDatagovByEntityBySpecificDate);
14
15 module.exports = router;
16
```

Ilustración 46 Router Datos del gobierno

e. Recomendaciones

Controlador de recomendaciones

El en archivo controlador de recomendaciones obtenemos la conexión a la base de datos mediante el contexto de ejecución y se lo pasamos al modelo. En el controlador, manejamos los datos obtenidos mediante el modelo o los errores que hubo en la petición o con la consulta sobre la base de datos.

Funciones de controlador

- Obtener todas las recomendaciones.
- Agregar nuevas recomendaciones.
- Obtener una recomendación por id.
- Actualizar una recomendación por id.
- Eliminar una recomendación por id.
- Obtener una recomendación por nivel.

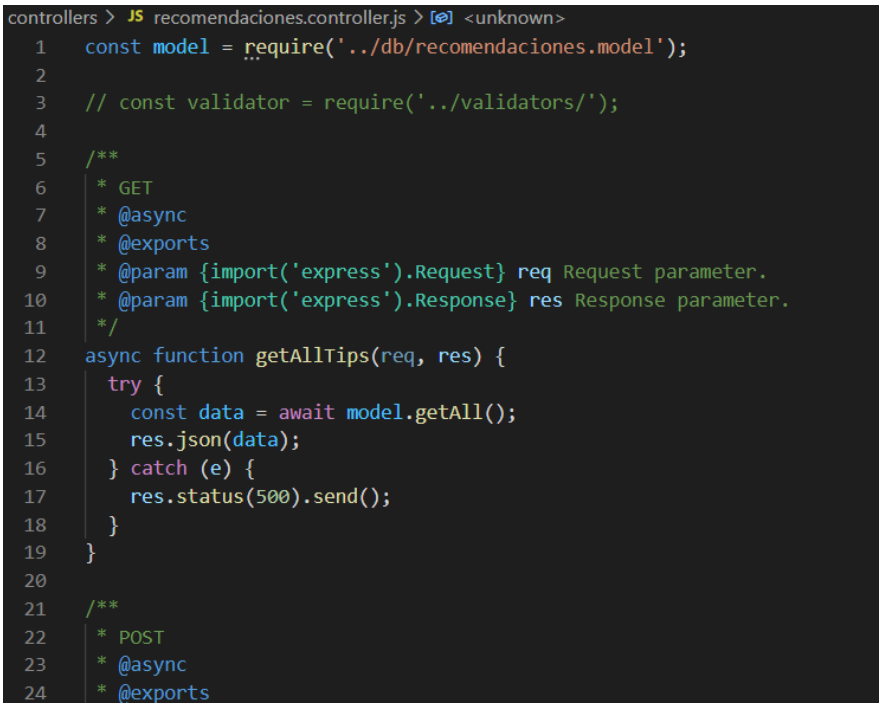


Ilustración 47 Controlador de recomendaciones

Modelo de recomendaciones

En el archivo modelo de recomendaciones llevamos a cabo la conexión a la base de datos y obtenemos recomendaciones por id o por nivel, actualizamos, borramos, e insertamos nuevas.

Funciones del modelo

- Obtener todas las recomendaciones.
- Agregar nuevas recomendaciones.
- Obtener una recomendación por id.
- Actualizar una recomendación por id.
- Eliminar una recomendación por id.
- Obtener una recomendación por nivel.

db > recomendaciones.model.js > postNew > <function>

```
const { getConnection } = require('../config/dbConfig');

async function getAll() {
  const connection = await getConnection();

  return new Promise((resolve, reject) => {
    connection.query('SELECT * FROM recomendaciones', (err, results) => {
      if (err) {
        reject(err);
        return;
      }
      resolve(results);
    });
  });
}

async function postNew(data) {
  const connection = await getConnection();

  return new Promise((resolve, reject) => {
    const q = `INSERT INTO recomendaciones (nivelRecomendaciones, descripcion) VALUES (${data.nivel}, '${data.descripcion}')`;

    connection.query(q, (err, results) => {
      if (err) {
```

Ilustración 48 Modelo recomendaciones

Rutas

- Obtener todas las recomendaciones.
- Agregar nuevas recomendaciones.
- Obtener una recomendación por id.
- Actualizar una recomendación por id.
- Eliminar una recomendación por id.
- Obtener una recomendación por nivel.

routes > JS recomendaciones.router.js > ...

```
const express = require('express');

const controller = require('../controllers/recomendaciones.controller');

const router = express.Router();

router.get('/', controller.getAllTips);
router.post('/', controller.postNewTip);
router.get('/id/:id', controller.getTipByID);
router.put('/id/:id', controller.putTipByID);
router.delete('/id/:id', controller.deleteTipByID);
router.get('/nivel/:nivel', controller.getTipByNivel);

module.exports = router;
```

Ilustración 49 Router de recomendaciones

f. Mapa

Controlador de Mapa

El en archivo controlador de mapa obtenemos la conexión a la base de datos mediante el contexto de ejecución y se lo pasamos al modelo. En el controlador, manejamos los datos obtenidos mediante el modelo o los errores que hubo en la petición o con la consulta sobre la base de datos.

Funciones de controlador

- Obtener registros de pruebas en un radio indicado.

controllers > JS mapa.controller.js

```
const model = require('../db/mapa.model');
const { executionContext } = require('../db/executionContext');
/**
 * GET
 * @async
 * @exports
 * @param {import('express').Request} req Request parameter.
 * @param {import('express').Response} res Response parameter.
 */
async function getMapaRadio(req, res) {
  executionContext((context) => {
    const { connection } = context;
    const { radio } = req.params;
    model.getByRadio(connection, radio)
      .then((data) => {
        res.statusCode = 200;
        res.send(data);
      })
      .catch((err) => {
        if (Object.prototype.hasOwnProperty.call(err, 'sqlMessage')) {
          res.status(400).send(err.sqlMessage);
        } else {
          res.status(500).send(err);
        }
      });
```

Ilustración 50 Controlador mapa

Modelo de Mapa

En el archivo modelo de mapa llevamos a cabo la conexión a la base de datos y obtenemos un conteo de las pruebas que tengan como escrutinio 2, 3, o 4 por cada clúster que se formaba dependiendo al a los decimales que se utilizaban de acuerdo con la longitud a la latitud.

Funciones del modelo

- Obtener registros de pruebas en un radio indicado.

```
db > JS mapa.model.js > getByRadio > <function>
1  const QUERY_GET_BY_RADIO = 'SELECT actualizaciones.escrutinio, ROUND(usuarios.latitud, ?
2
3  async function getByRadio(connection, radio) {
4    return new Promise((resolve, reject) => {
5      let decimals;
6      switch (radio) {
7        case 'small':
8          decimals = 3;
9          break;
10       case 'medium':
11         decimals = 2;
12         break;
13       case 'large':
14         decimals = 1;
15         break;
16       default:
17         decimals = 6;
18     }
19     connection.query(QUERY_GET_BY_RADIO,
20       [decimals, decimals, decimals, decimals], (err, results) => {
21       if (err) {
22         reject(err);
23         return;
24       }
25       resolve(results);
```

Ilustración 51 Modelo mapa

Rutas

- Obtener registros de pruebas en un radio indicado.

```
routes > JS mapa.router.js > ...
1  const express = require('express');
2
3  const controller = require('../controllers/mapa.controller');
4
5  const router = express.Router();
6
7  router.get('/:radio', controller.getMapaRadio);
8
9  module.exports = router;
10
```

Ilustración 52 Router mapa

3. Resultados del trabajo profesional

I. Pantalla de inicio (*Splash Screen)*

Ilustración 53 Pantalla de inicio

En este primer producto de la aplicación es la pantalla que se carga al abrir la aplicación. Lo que se hace en esta pantalla es pedir los permisos necesarios al usuario (ubicación), validar si es la primera vez que se abre la aplicación y abrir la pantalla de presentación o abrir la pantalla de registro o de inicio de sesión dependiendo si hay una sesión activa o no.

Este primer producto implicó en lo profesional desarrollar el archivo XML en el que se define a Interfaz de Usuario de la pantalla y su respectiva actividad en Kotlin, investigar el manejo de permisos en tiempo de ejecución (*Runtime Permissions)* y el uso de *SharedPreferences* para almacenar datos de forma local.

II. Pantalla de Presentación

Ilustración 54 Pantalla de presentación

La pantalla de presentación es la que solo se muestra al usuario solo la primera vez que abre la aplicación para hacerles saber el objetivo y principales funcionalidades de esta. En esta pantalla se puede navegar a la pantalla de Registro de usuario o Inicio de Sesión dependiendo de lo que el usuario indique.

Esta pantalla fue de las más sencillas debido a que en lo profesional solo implicó desarrollar el archivo XML con la descripción de la Interfaz de Usuario y su respectiva Actividad en Kotlin.

III. Pantalla de Registro de Usuario

Ilustración 55 Pantalla de registro

El registro de usuario se muestra mientras el usuario no se haya registrado en la aplicación. La funcionalidad de esta pantalla es permitir a nuevos usuarios registrarse de forma anónima para ser capaces de utilizar la aplicación. Desde esta pantalla solo se puede dirigir a la pantalla de inicio de sesión ya que es necesario obtener el token del usuario después de su registro.

Desde lo profesional esta pantalla implicó desarrollar el archivo XML con la descripción de la Interfaz de Usuario y su respectiva actividad en Kotlin, utilizar un *ViewModel* para hacer uso del repositorio de usuarios (cliente Retrofit). La integración del *ViewModel* y el repositorio se hizo utilizando el patrón arquitectónico MVVM lo que implicó investigación relaciona con este patrón y el uso de Retrofit.

IV. Pantalla de Inicio de Sesión

Ilustración 56 Pantalla de inicio de sesión

El inicio de sesión se muestra mientras el usuario no tenga una sesión activa, es decir, que no esté autenticado. La funcionalidad de esta pantalla es permitir a los usuarios iniciar sesión de forma anónima para autenticarse y ser capaces de utilizar la aplicación. Desde esta pantalla solo se puede dirigir a la pantalla del menú principal.

Desde lo profesional esta pantalla implicó desarrollar el archivo XML con la descripción de la Interfaz de Usuario y su respectiva actividad en Kotlin, utilizar un *ViewModel* para hacer uso del repositorio de autenticación (cliente Retrofit). La integración del *ViewModel* y el repositorio se hizo utilizando el patrón arquitectónico MVVM lo que implicó investigación relaciona con este patrón y el uso de Retrofit. También implicó el uso *SharedPreferences* para poder almacenar localmente el folio y el token de usuario.

V. Pantalla de Menú

Ilustración 57 Pantalla de menú principal

Mientras la sesión del usuario esté activa después de la pantalla de inicio se redirigirá a esta pantalla. Esta pantalla permite al usuario tener un menú de interacción sencillo para poder hacer uso de los servicios que se ofrecen con esta aplicación: mapa de contagios, pruebas, novedades, recomendaciones, hospitales cercanos, estadísticas, ajustes y avisos de privacidad.

Esta pantalla al igual que la de presentación fue de las más sencillas desde una perspectiva profesional debido a que solo implicó el desarrollo del archivo XML con la descripción de la Interfaz de Usuario y su respectiva actividad en Kotlin y la navegación a todos los servicios de la aplicación.

VI. Pantalla introducción test

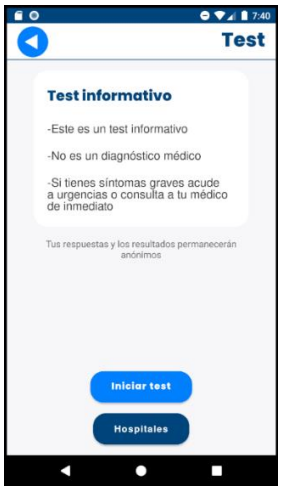


Ilustración 58 Pantalla de introducción a prueba

Cuando el usuario de clic en la opción de prueba/test del menú se mostrará esta pantalla en caso de que no se haya realizado una prueba previamente con ese dispositivo. El usuario tiene las opciones de consultar los hospitales o de iniciar una prueba nueva.

Desde lo profesional esta pantalla implicó desarrollar el archivo XML con la descripción de la Interfaz de Usuario y su respectiva actividad en Kotlin, utilizar fragmentos para encapsular el comportamiento prueba en una única actividad y de *SharedPreferences* para saber si ya se ha hecho alguna prueba previamente.

VII. Pantalla preguntas test

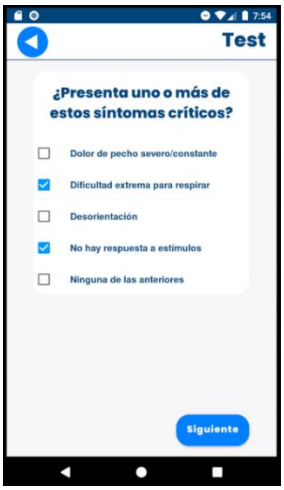


Ilustración 59 Pantalla de prueba

Cuando el usuario da clic en iniciar test se cambia a esta pantalla en la que se muestran preguntas con sus opciones para ser seleccionadas. Se tienen botones para controlar el flujo de la prueba, en la primera pregunta se oculta el botón de anterior y en la última se sustituye el botón de siguiente por el de terminar.

Las preguntas se toman de un JSON que contiene un arreglo con las preguntas, sus opciones y valores para saber que opciones han sido seleccionadas. La prueba está preparada para mostrar cualquier número de preguntas con tantas opciones como se desee. Al terminar la prueba las respuestas se registran en la base de datos y se obtiene el escrutinio. Esta pantalla también requirió el uso del patrón *ViewModel* y de un repositorio de *Retrofit*.

VIII. Pantalla resultados prueba

Ilustración 60 Pantalla de resultados de prueba

Al recibir el escrutinio después de enviar la prueba se muestran los resultados de acuerdo con el escrutinio, existen 4 tipos de resultados que pueden ser mostrados al usuario. Los escrutinios se muestran en su respectivo color y se muestran las recomendaciones que se tiene para cada nivel. Se tiene un botón que te lleva a la pantalla de Recomendaciones y en caso de que el escrutinio sea 4 se tiene un botón que te lleva a hospitales.

IX. Pantalla repetir test

Ilustración 61 Pantalla de repetir prueba

Cuando se da clic en la opción de prueba del menú principal se muestra esta pantalla en caso de que ya se haya realizado una prueba en el dispositivo. Se muestra el resultado con sus recomendaciones y la fecha en la que se hizo la prueba. También se tiene un botón para hacer la prueba de nuevo.

Con esta pantalla se requirió el uso de *SharedPreferences* para saber si se ha hecho una prueba anteriormente, la fecha en la que se realizó y cuál fue el resultado.

X. Pantalla novedades

Ilustración 62 Pantalla novedades

Cuando se da clic en opción de novedades del menú se abre esta pantalla, la cual permite ver los datos del gobierno que se tienen para cada estado y la fecha en la que se registraron esos datos. Con un *Spinner* el usuario puede seleccionar cual es el estado del que desea ver los datos. Se cuenta con dos botones para redirigir al usuario a otras dos páginas externas donde puede revisar los datos a escala nacional o global.

Al igual que otras pantallas esta pantalla implicó desarrollar el archivo XML con la descripción de la Interfaz de Usuario y su respectiva actividad en Kotlin, utilizar un *ViewModel* para hacer uso del repositorio de Datos del Gobierno (cliente Retrofit). La integración del *ViewModel* y el repositorio se hizo utilizando el patrón arquitectónico MVVM.

XI. Pantalla estadísticas

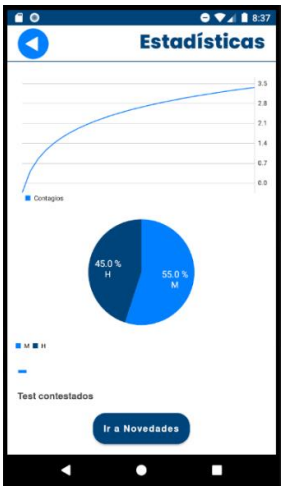


Ilustración 63 Pantalla de estadísticas

En esta pantalla la aplicación muestra las estadísticas relacionadas con las pruebas o test realizados dentro de la línea. La gráfica en la parte superior muestra el número de posibles casos (Pruebas con escrutinio 4) en el último mes. Por otro lado, la gráfica de pastel muestra el total de pruebas sin importar el escrutinio según el sexo de usuario H (hombre) y M (mujer).

Al igual que los otros productos, para esta actividad fue necesario desarrollar el archivo XML con la descripción de la Interfaz de usuario y su respectiva actividad en Kotlin. Para mostrar las gráficas fue necesario investigar que librerías existen y cómo podemos utilizarlas, al final nos decidimos por la librería código abierto *MPAndroidChart*.

Para poder obtener los datos de las gráficas nos comunicamos con el repositorio de pruebas utilizando el cliente de *Retrofit* relacionadas con esa tabla de la base de datos y lo integramos con la aplicación utilizando el modelo arquitectónico MVVM.

XII. Pantalla Recomendaciones/Tips

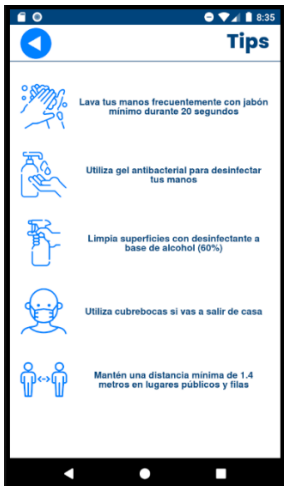


Ilustración 64 Pantalla de recomendaciones

En la pantalla de recomendaciones o tips se muestran las recomendaciones generales que indican medios oficiales. Para esta parte de la aplicación se requirió desarrollar el XML con la descripción de la Interfaz de usuario y su respectiva actividad en Kotlin. En esta vista no nos comunicamos con los servicios web si no que, se tienen definidos las recomendaciones en la aplicación y se despliegan utilizando un *RecyclerView*.

XIII. Pantalla Ajustes

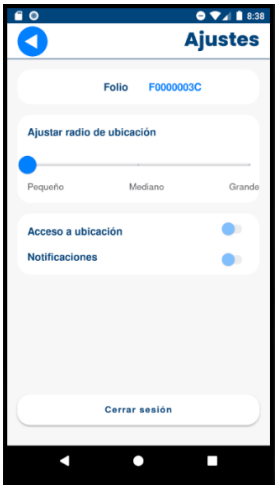


Ilustración 65 Pantalla de ajustes

En la pantalla de ajustes se muestra el folio del usuario actual en la aplicación y se facilitan dos funcionalidades. La primera es el ajuste del radio de las zonas de contagio a mostrar en el mapa y el cierre de sesión. Al igual que las otras pantallas fue necesario desarrollar el XML con la descripción de la Interfaz de usuario y su respectiva actividad en Kotlin. Esta pantalla no se comunica con los servicios web. El radio lo guardamos localmente utilizando *SharedPreferences* y el cierre de sesión implica borrar los datos guardados de forma local relacionados con la sesión (el token).

XIV. Pantalla acerca de

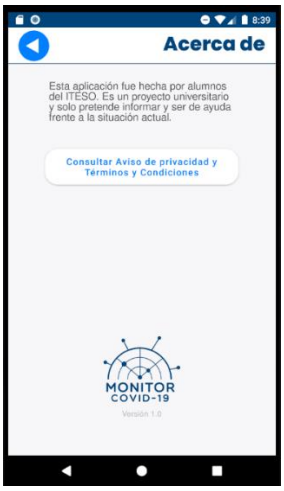


Ilustración 66 Pantalla Acerca De

En la pantalla de acerca de se muestra información relevante de la aplicación como el propósito y la versión de esta. Además, existe un botón con el que mostramos al usuario el archivo PDF que contiene el aviso de privacidad y los términos y condiciones que conlleva el uso de la aplicación. Al igual que con las otras pantallas fue necesario desarrollar el archivo XML con la descripción de la Interfaz de usuario y su respectiva actividad en Kotlin. Esta pantalla no se comunica con los servicios web.

XV. Pantalla mapa

Ilustración 67 Pantalla mapa zonas de riesgo

En la pantalla de mapa se muestran las zonas de riegos basándonos en las pruebas realizadas por los usuarios de la aplicación. El radió de estas áreas de riesgo puede ser modificado en la pantalla de ajustes. Las zonas de contagio pueden ser de dos colores, rojas o amarillas. En las zonas rojas agrupamos a todos los usuarios que tiene pruebas con un escrutinio igual a cuatro, en las amarillas aquellos usuarios con escrutinios 2 o 3.

Esta pantalla no solo requirió el desarrollo del XML con la descripción de la Interfaz de usuario, también fue necesario investigar el SDK de *GoogleMaps* para Android y el uso de estas librerías para poder hacer uso de mapas. Esta pantalla se comunica con uno de los servicios web desarrollados que nos ofrece una lista de zonas de contagio a ser dibujadas en el mapa. Para comunicarnos con los servicios web del mapa lo hacemos mediante un cliente *Retrofit* e integramos el uso de cliente utilizando el patrón arquitectónico MVVM.

XVI. Pantalla hospitales

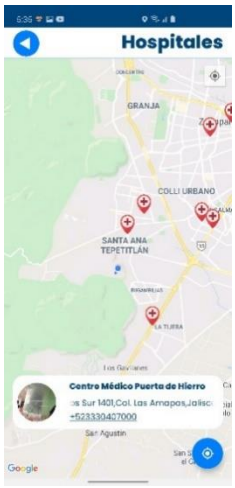


Ilustración 68 Pantalla mapa de hospitales

En esta pantalla se muestran los hospitales a la ubicación del usuario en un radio de 10 km. Además de mostrar el marcador o pin de los hospitales, al dar clic a este se muestra una ventana de información sobre el hospital seleccionado.

Al igual que la pantalla de zona de contagios, para esta actividad utilizamos el SDK de *Google Maps* para el mapa. Además de utilizar el SDK de Google utilizamos una API de Yelp que usamos para buscar los hospitales cercanos.

Para comunicarnos con los servicios web que ofrece Yelp creamos un cliente con Retrofit y definimos los modelos de dichos servicios. El uso del repositorio lo integramos en la aplicación utilizando el patrón arquitectónico MVVM.

4. Reflexiones del alumno o alumnos sobre sus aprendizajes, las implicaciones éticas y los aportes sociales del proyecto

- Aprendizajes profesionales

Isaac Cabrera

Pienso que una de las competencias más importantes que pude desarrollar durante este proyecto fue el trabajo en equipo multidisciplinario teniendo contacto con personas enfocadas con el diseño y los servicios web de la aplicación. Desde lo profesional implico hacer uso de competencias que desarrollé en la carrera pero que en el proyecto pude poner a prueba relacionadas con el desarrollo aplicaciones móviles y servicios web.

Como mencioné anteriormente el trabajo en equipo fue parte clave, así como la planeación del proyecto. Creo que una competencia importante fue lidiar con cambios imprevistos en el desarrollo ocasionados por diferentes motivos como ausencia de servicios web o falta de presupuesto para el uso de ciertas herramientas. Otra competencia muy importante desde mi punto de vista fue la capacidad de resolución de problemas e investigación.

El haber hecho este proyecto me ayudo a entender la problemática social y económica que representan fenómenos como lo es la pandemia. Creo que para poder ofrecer un servicio de calidad fue necesario adentrarnos precisamente a los problemas que causó y está causando el Coronavirus en México y el mundo.

Durante el proyecto fueron surgiendo nuevos retos en los que me vi puesto a prueba. Por ejemplo, la investigación y uso de librerías que nunca había utilizado, el uso de nuevas tecnologías como CPANEL y el aprender nuevos patrones de diseño como el modelo Vista Controlador aplicado en la refactorización del *Backend*.

Creo que el proyecto me permitió aplicar mis conocimientos técnicos para poder ayudar a resolver una problemática actual que afecta a millones de personas en el mundo. Los conocimientos que tuve que aplicar fueron diversos como el desarrollo de aplicaciones móviles, servicios web y bases de datos.

Alessandro Pallaro

En la creación de la aplicación de MonitorCovid pude reforzar mis conocimientos en Android Studio, aprender diferentes metodologías de diseño como lo es MVVM, conocer como es la infraestructura de una aplicación móvil con un *Backend* en CPanel, planificar una Rest API y desarrollarla para posteriormente conectarla con la aplicación. En cuanto a competencias más genéricas, pude aprender cómo es la comunicación con compañeros de trabajo de otras áreas cómo *Backend* o diseño y presentarle avances al cliente, que en este caso fue el profesor quien tomó ese papel.

Las disciplinas en las que pienso que desarrollé competencias fueron la administración de proyectos, el desarrollo móvil, el desarrollo de servicios web y la comunicación oral y escrita. Las competencias que adquirí en estas áreas fueron el planificar un proyecto de gran tamaño y tratar de apegarse al plan y ajustar el plan cuando sea necesario, el patrón de diseño MVVM y conexiones por medio de *Retrofit*, como hacer una API Rest y por último el comunicarme con mis compañeros para dar a entender mis ideas y para hacer los reportes solicitados.

Considero que lo que más se puso a prueba en este proyecto fue el saber estimar la duración que tendría cada tarea que debíamos realizar para llegar al producto final, ya que en varias ocasiones al realizar el plan del trabajo dábamos un tiempo estimado para alguna tarea pero al momento de hacerlo nos dábamos cuenta de que necesitábamos más tiempo porque la tarea es más compleja de lo que creíamos. También durante todo el proyecto las habilidades de programación fueron puestas a prueba para darle las funcionalidades requeridas a la aplicación.

Aprendí como colaborar con mis compañeros para crear un producto complejo desde la etapa de diseño hasta su desarrollo e implementación. Pude conocer como distintos formatos que presentan del área de diseño que son necesarios para desarrollar una aplicación, por ejemplo, los escenarios y el flujo. También aprendí de mis compañeros otras formas de trabajar y organizarse que me podrían ser útiles en un futuro.

- Aprendizajes sociales

Isaac Cabrera

El desarrollo de este proyecto me permitió utilizar mi creatividad y conocimientos para poder ofrecer un producto o servicio que contribuya a resolver una problemática social. Aprendí que para poder hacer esto es necesario una investigación completa de la problemática para poder identificar cual es el verdadero problema y ayudar a solucionarlo, no enfocarnos en problemáticas menores que probablemente no contribuyan a la solución del problema principal. Creo que es este caso nosotros tratamos de resolver la desinformación de muchas personas en México relacionada con la situación actual de la pandemia.

Ahora que el proyecto está prácticamente terminado, me doy cuenta de todo lo implica hacer uno, desde la investigación de la problemática a resolver, los requerimientos del producto o servicio, el diseño de la solución y su implementación.

Con el desarrollo de este proyecto pude darme cuenta de la capacidad que tengo para innovar y hacer uso de la tecnología para abonar a la solución de problemáticas en el ámbito social. Si bien muchas veces se ve al sector tecnológico como algo separado del ámbito social, creo que si podemos quitar esa barrera y aplicar la tecnología en todo de tipo de situaciones podemos ser capaces de ayudar a resolver problemáticas de alto impacto para las personas.

Hablando específicamente de este proyecto (Monitor Covid) creo que el proyecto puede beneficiar a un gran número de mexicanos. Esto porque el propósito de la aplicación es mantener informada a la población en México sobre la situación actual de la pandemia y ayudarlos a cuidarse y mantener la tasa de contagios controlada. Al tener el número de contagios estable ayuda a la economía mexicana porque los negocios pueden seguir abiertos y los servicios de salud no se colapsan por el alto número de contagios.

Lo que más me motiva del proyecto es ver cómo la tecnología aplicada a una problemática social puede ser replicada en muchos otros casos.

Después de varios meses de pandemia podemos darnos cuenta de que este problema puede seguir por mucho más tiempo de lo esperado, por lo que la aplicación puede crecer y no solo enfocarse en México si no eventualmente cubrir un área mucha más grande como América o el mundo.

Este proyecto me ayudó a abrir los ojos y darme cuenta del impacto que puedo lograr haciendo uso de mis conocimientos y aplicarlos a los problemas que existen en México y el mundo. Los resultados obtenidos me motivan a seguir contribuyendo a la resolución de problemas sociales mediante el uso de la tecnología.

Alessandro Pallaro:

La pandemia no solo ha afectado a la gran cantidad de personas que han fallecido a causa del virus, sino que prácticamente todos somos afectados, unos más que otro pero todos estamos siendo afectados. En lo económico muchos negocios terminaron en bancarrota debido a que no pudieron mantenerse durante el parón, y otros tuvieron que reducir el personal para seguir adelante. Muchas personas que dependían de esos empleos para proveer a sus hogares tendrán que buscar otras alternativas. Otras personas que tienen alguna enfermedad crónica no han podido acceder a los servicios de salud al ser una población de riesgo. Con la aplicación Monitor Covid se busca ayudar a que la pandemia termine lo antes posible y a controlar el total de contagios así que de cierta forma estaría ayudando a toda la comunidad.

Al estar en contacto con todas las estadísticas que se tienen sobre el covid-19 durante el desarrollo de la aplicación pude entender la gravedad de la situación y pienso que de esta misma forma la aplicación que desarrollamos puede ayudar a que la gente sea más consciente del problema. El desarrollo móvil cada vez está más presente en la vida cotidiana de todos y aplicaciones de este tipo pueden causar un gran impacto para el bien común.

Pienso que de la misma forma que se utilizó el desarrollo móvil en este proyecto, se podría aplicar para atender muchas otras problemáticas sociales como lo que se planea hacer con la app de la red de sensores del bosque de la primavera, la cuál ayudara a la conservación del bosque de la primavera.

- Aprendizajes éticos

Isaac Cabrera

Creo que desde el punto de vista ético nosotros como desarrolladores tenemos hacer un buen uso de los datos de los usuarios, no lucrar con ellos. Es bien sabido que grandes empresas tecnológicas han sido enjuiciadas en Estados Unidos por lucrar con los datos de sus usuarios y venderlos a distintas empresas. Pienso que hay que respetar la privacidad de los usuarios, es por lo que se piden los permisos que se consideran peligrosos (ubicación) en tiempo de ejecución y se anexan los términos y condiciones de uso en una de las pantallas de la aplicación.

A pesar de no participar en la creación del documento de términos y condiciones, aprendí que es necesario definir uno para proteger a los usuarios, pero también a nosotros. Se tiene que explicar de buena forma al usuario para que son necesarios los datos que le solicitamos y qué haremos con ellos.

Como ya mencioné en otros apartados, el haber participado en este proyecto me motiva a hacer uso de la tecnología para ayudar a las personas y tratar de resolver problemáticas en el ámbito social que aquejan a México y el mundo. Después de este proyecto, tengo pensado analizar algunas de las problemáticas que vivimos hoy en día y definir un proyecto en el que con la ayuda de la tecnología resolvamos o tratemos de resolver dichas problemáticas.

Alessandro Pallaro

Tomé la decisión de inscribirme al PAP de vida digital ya que el tema de ciudades inteligentes es algo que me interesa ya que involucra en gran medida mi profesión y promueve un desarrollo sostenible mejorando la calidad de vida de los ciudadanos. Pienso que esta decisión me trajo consecuencias favorables ya que me llenado de varias ideas de cómo mejorar nuestra ciudad con el uso de las tecnologías.

Aunque casi no me tocó participar en el proyecto de redes WSN estuve presente en las reuniones en la que mis demás compañeros presentaban sus avances. Esto me ayudó a darme una idea de cómo mi carrera puede interactuar con otras, como electrónica, mecánica o diseño, para lograr realizar un proyecto más completo y de mayor impacto.

Esta experiencia despertó curiosidad en mi sobre a qué otros proyectos podrían aportar algo profesionalmente y que traigan un beneficio social.

- Aprendizajes en lo personal

Isaac Cabrera

Personalmente creo que el haber estado en este PAP me ayudó a darme cuenta de la capacidad que tengo para el trabajo en equipo y ser líder de un grupo. A lo largo de la universidad muy pocas cosas veces fungí como líder en algún proyecto o trabajo, sin embargo, para el PAP fue necesario dejar eso de lado y me hacerme cargo del equipo de *Frontend* móvil. Otro punto importante del que pude darme cuenta fue mi capacidad para solucionar problemáticas en proyectos como cambios en la planeación y buscar alternativas en el desarrollo debido a un presupuesto limitado.

Creo que algo que me costó fue la comunicación con otros equipos como *Backend* y Diseño, no es que la comunicación era mala en el sentido de malos tratos o falta de profesionalidad, si no que no era rápida y pocas veces dábamos seguimiento al tema.

Creo que a pesar de que el trabajo multidisciplinar fue uno de los mayores retos a lo largo del PAP, fue lo que más me gusto porque pude darme cuenta de cómo distintos campos se unen para crear algo, en este caso la aplicación. Pienso que aprendí mucho del equipo de *Backend* y Diseño porque siempre fue muy valiosa su retroalimentación a la hora de desarrollar cosas que no se consideraron en un principio o se redefinieron durante el desarrollo.

El haber participado en este proyecto me hizo pensar que es lo que quiero para mi proyecto de vida y después de meditar varios días me di cuenta de que además de buscar una realización personal, quiero poder ayudar a las personas aplicando mis conocimientos y utilizando nuevas tecnologías.

Alessandro Pallaro

Este proyecto me ha enseñado que soy capaz de hacer grandes cosas cuando trabajo en equipo, pienso que el resultado obtenido en la app del Covid junto con mi compañero

Isaac y los equipos de backend y de diseño es algo de que sentirse orgullosos. Saber que esto puede ayudar a los demás en cierto modo motiva a seguir haciendo proyectos de este tipo. No solo preocuparme por el bien común y hacer algo al respecto para resolver alguna problemática, me hace replantearme a que es a lo que me quiero dedicar en un futuro y ser una persona más altruista desde mi profesión.

Honestamente antes de este PAP en algunas ocasiones llegaba a menospreciar el trabajo de otras carreras, sin embargo al trabajar en conjunto con esas carreras y utilizar productos que ellos realizaron nos hizo llegar a un producto final de mayor calidad en de forma más fácil.

En cuanto a mi proyecto de vida, el PAP vida digital me ha hecho replanteármelo seriamente, ya no sólo quiero tener una carrera profesional que me brinde estabilidad económica, ahora también quiero buscar aportar al bien común con mis habilidades profesionales.

5. Conclusiones

A lo largo del proyecto tuvimos que realizar varios cambios que afectaron el plan de trabajo definido al inicio del PAP. El plan de trabajo tuvo que ser actualizado porque fue necesario realizar las pantallas de la aplicación desde cero porque lo que se tenía del semestre pasado no contaba con buenas prácticas y refactorizar el *Backend* de la aplicación para poder integrar los servicios web de forma exitosa. Las actividades relacionadas con el *Frontend* de la aplicación tuvieron que ponerse en espera durante cuatro semanas mientras se terminaba de refactorizar el *Backend*.

A pesar de que el tiempo de desarrollo se incrementó debido a la refactorización, el equipo respondió de buena forma y pudimos cumplir con el nuevo plan de trabajo. Independientemente de que no se pudo iniciar el desarrollo de aplicación de la red de sensores, creemos que el objetivo se cumplió porque pudimos desarrollar la aplicación Monitor Covid utilizando las mejores prácticas. Aunque pudimos hacer un buen trabajo para la primera versión de la aplicación, creemos que todavía existen puntos de mejora.

La prueba la implementamos para que fuera capaz de mostrar cualquier cantidad de preguntas que se le mande desde *Backend* con cualquier cantidad de opciones por pregunta, de esta forma se logra tener una prueba que sea más dinámica y fácil de actualizar. Sin embargo, en el *Backend* no se cuenta con un servicio que proporcione las preguntas, ni un espacio en la base de datos para las preguntas, por lo que en la aplicación este servicio quedó simulado mediante un JSON y una oportunidad de mejora sería implementar ese servicio y modificar el repositorio de pruebas para que las respuestas de la prueba se manden de igual forma.

Otro punto que no es una mejora a la aplicación, pero si un buen complemento, es un panel administrativo. Este panel puede ser una página web en la que se proporcione una UI intuitiva que ayude a los alumnos del PAP a administrar la base de datos. Para dicha administración sería necesario permitir al usuario realizar operaciones CRUD en cada tabla de la base datos. Esto sería útil para actualizar los teléfonos de contacto y recomendaciones que se muestran en la aplicación.

6. Bibliografía

1&1 IONOS Inc. (2020, 2 noviembre). *Programación funcional*. IONOS Digital Guide. https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/programacion-funcional/

Atlassian. (2020). *Jira - Software de seguimiento de proyectos e incidencias*. https://www.atlassian.com/es/software/jira

B., G. (2019, 13 mayo). ¿Qué es GitHub y para qué se utiliza? Tutoriales Hostinger. https://www.hostinger.mx/tutoriales/que-es-github/

Feliciano, C. (2019, 1 febrero). *¿Qué es el desarrollo de aplicaciones móviles?* INVID. https://invidgroup.com/es/que-es-el-desarrollo-de-aplicaciones-moviles/

Google. (2019, 27 diciembre). *Kotlin overview*. Android Developers. https://developer.android.com/kotlin/overview

Naeem, T. (2020, 3 noviembre). *REST API: definición, funcionamiento, beneficios y principios de diseño*. Astera. https://www.astera.com/es/type/blog/rest-api-definition/

Ortega, J. M. M. (2020, 21 abril). *La arquitectura MVVM y sus componentes*. OpenWebinars.net. https://openwebinars.net/blog/la-arquitectura-mvvm-y-sus-componentes/

Pérez, J., & Gardey, A. (2010, 1 enero). *Definición de XML*. Definición.de. https://definicion.de/xml/

ProyectosAgiles.org. (2018, 9 octubre). *Qué es SCRUM*. Proyectos Ágiles. https://proyectosagiles.org/que-es-scrum/

Readfearn, G. (2020, 1 julio). *How did coronavirus start and where did it come from? Was it really Wuhan's animal market?* The Guardian. https://www.theguardian.com/world/2020/apr/28/how-did-the-coronavirus-start-where-

did-it-come-from-how-did-it-spread-humans-was-it-really-bats-pangolins-wuhan-animal-market

Ríos, J. (2019, 19 mayo). *El Bosque de la Primavera ha perdido más patrimonio ecológico en tres décadas que en 140 mil años, afirman especialistas*. La Red Universitaria Jalisco. http://www.udg.mx/es/noticia/bosque-primavera-ha-perdido-mas-patrimonio-ecologico-tres-decadas-140-mil-anos-afirman

Square Inc. (2013, 1 enero). *Retrofit*. square.github.io. https://square.github.io/retrofit/

Vázquez, M. (2018, 18 marzo). *PROGRAMACIÓN ORIENTADA A OBJETOS*. Inter Blog. https://www.lainter.edu.mx/blog/2018/03/18/programacion-orientada-a-objetos/