



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE APIZACO

DESARROLLO APP/MÓVIL PROYECTO NATIVISPROJEX

TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

**PARA OBTENER EL TÍTULO PROFESIONAL DE:
INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN Y
COMUNICACIONES**

PRESENTA:

XIMENA CHACON ROJAS

NOMBRE DEL ASESOR:

EDUARDO SANCHEZ LUCERO



APIZACO, TLAX., FEBRERO 2024



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO

Instituto Tecnológico de Apizaco
Sistemas y Computación

Apizaco, Tlaxcala, **07/septiembre /2023**

OFICIO No: SC/OPV/186/23

ASUNTO: Liberación de proyecto
para titulación integral

MARTÍN ROJAS RAMÍREZ
JEFE DEL DEPTO. DE DIVISIÓN DE ESTUDIOS PROFESIONALES
P R E S E N T E.

POR ESTE MEDIO LE INFORMO QUE HA SIDO LIBERADO EL SIGUIENTE PROYECTO PARA LA TITULACIÓN INTEGRAL.

NOMBRE DEL EGRESADO:	XIMENA CHACÓN ROJAS
CARRERA:	INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
NO. DE CONTROL:	18370930
NOMBRE DEL PROYECTO:	DESARROLLO APP/MÓVIL PROYECTO NATIVISPROJEX
PRODUCTO U OPCIÓN:	TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

AGRADEZCO DE ANTEMANO SU VALIOSO APOYO EN ESTA IMPORTANTE ACTIVIDAD PARA LA FORMACIÓN PROFESIONAL DE NUESTROS EGRESADOS.

ATENTAMENTE

*Excelencia en Educación Tecnológica®
Pensar para Servir, Servir para Triunfar®*

ALAN AUGUSTO GALLEGOS CUELLAR
JEFE DEL DEPTO. DE SISTEMAS Y COMPUTACIÓN

c.c.p. División de Estudios Profesionales
c.c.p. Depto. de Servicios Escolares
c.c.p. Archivo
AAGC/mrol*


SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL
DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
**DEPARTAMENTO DE
SISTEMAS Y COMPUTACIÓN**


EDUARDO SÁNCHEZ LUCERO
ASESOR INTERNO



Carretera Apizaco-Tzompantepec, esquina con Av. Instituto Tecnológico S/N,
Conurbado Apizaco-Tzompantepec, Tlaxcala, Mex. C.P. 90491 Tel. (241) 4172010 Ext. 115,
e-mail: sistemas@apizaco.tecnm.mx tecnm.mx | apizaco.tecnm.mx



2022 Flores
Año de Magón
PRESENCIA DE LA REVOLUCIÓN MEXICANA



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO

Instituto Tecnológico de Apizaco
División de Estudios Profesionales

Apizaco, Tlax., 26/enero/2024
No. OFICIO: DEP-CT/040/2024

ASUNTO: AUTORIZACIÓN DE IMPRESIÓN

XIMENA CHACON ROJAS
EGRESADA DE LA CARRERA DE
INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
CON NÚMERO DE CONTROL: 18370930
P R E S E N T E

Por este medio, me permito informar a usted que por aprobación de la comisión revisora asignada para valorar el trabajo que mediante la opción **TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL** cuyo tema es: **"DESARROLLO APP/MÓVIL PROYECTO NATIVISPROJEX"**, conforme a lo establecido en el procedimiento para la obtención del Título Profesional en el Instituto Tecnológico, la División de Estudios Profesionales a mi cargo le otorga:

AUTORIZACIÓN DE IMPRESION

Debiendo entregar **3** empastados del mismo, apropiadamente encuadernados, a la brevedad, para presentar su acto de recepción profesional.

Sin otro particular por el momento, le envió un cordial saludo.

ATENTAMENTE

*Excelencia en Educación Tecnológica®
Pensar para Servir, Servir para Triunfar®*


MARTÍN ROJAS RAMÍREZ
JEFE DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES

c.p. archivo



Av. Instituto Tecnológico No. 418, San Andrés Ahuashuatepec, Municipio de Tzompantepec,
Tlaxcala, Mex. C.P. 90491 Tel. (241) 4172010 Ext. 304,
e-mail: dep_titulación@apizaco.technm.mx



2023
Francisco
VILLA

AGRADECIMIENTOS

A mis padres por haberme guiado y acompañado hasta este punto de mi vida, sin su ayuda y perseverancia no estaría en donde estoy; muchos de los logros y de la persona que soy hoy en día es gracias a ustedes. A pesar de que fueron estrictos, hoy se los agradezco porque sin eso probablemente no sería quien soy.

A mis hermanas por acompañarme todo este camino, muchas de las cosas igual las hago por ustedes, ustedes son mi segundo motor para seguir adelante.

A todos mis amigos, por ser mi apoyo, por escucharme y ayudarme en los momentos que pensaba en tirar la toalla, sin ustedes esto no hubiera sido lo mismo. Gracias por ser mi distracción muchas veces y por todos los momentos de felicidad. Cada uno de ustedes son una parte de lo que soy hoy y son una parte muy importante en esta vida.

Mi agradecimiento también va dirigido para la empresa ALLUXI, por darme la oportunidad de realizar mis residencias, en especial con todo el equipo de desarrollo de este proyecto ya que sin su ayuda no lo habría logrado y darme cuenta de que realmente quería dedicarme a este ramo de mi carrera.

Agradezco a la universidad por las oportunidades y conocimientos que me otorgo, en especial quiero agradecer a mis maestros de servicio social, los cuales siempre estuvieron dispuestos a enseñarnos y tenerle cariño a la carrera. Gracias por tantos conocimientos y enseñanzas de igual manera por los buenos ratos que pasábamos a pesar del trabajo.

Nada de esto estaría pasando sin ninguno de ustedes, gracias, por tanto.

RESUMEN

El proyecto de residencias consiste en el desarrollo de una aplicación móvil utilizando la tecnología React Native, la cual permitirá a los usuarios comprar y alquilar casas ecológicas. Esta aplicación tiene como objetivo fomentar el estilo de vida sostenible, brindando a los usuarios la oportunidad de vivir en hogares respetuosos con el medio ambiente.

La aplicación contará con diversas funcionalidades, como la búsqueda avanzada de casas ecológicas basada en ubicación, precio, tamaño y características específicas. Los usuarios podrán ver la información detallada de cada propiedad, incluyendo fotos, descripción, especificaciones técnicas y certificaciones ambientales.

Además, la aplicación permitirá a los usuarios contactar con los vendedores o arrendadores de las propiedades, agendar visitas para poder comprar alguna casa, hacer reservaciones, hacer las transacciones desde la aplicación, tener un apartado de red social en la cual podrán hacer publicaciones sobre las viviendas. También se incorporará un sistema de notificaciones para mantener a los usuarios informados sobre nuevas propiedades disponibles o cambios en las casas que han mostrado interés y notificaciones para la sección de red social.

La aplicación tomará ventaja de las capacidades de React Native para ofrecer una interfaz de usuario intuitiva y atractiva, compatible con dispositivos Android y iOS. Además, se implementarán características de seguridad, como la verificación de los perfiles de los usuarios y la protección de datos personales.

Tabla de contenido

CAPITULO I: GENERALIDADES DEL PROYECTO	9
1.1 INTRODUCCIÓN	9
1.2 DESCRIPCIÓN DE LA EMPRESA U ORGANIZACIÓN Y DEL PUESTO O ÁREA DEL TRABAJO DEL ESTUDIANTE	10
1.2.1 DESCRIPCIÓN DE LA EMPRESA	10
1.2.2 MISIÓN DE LA EMPRESA	10
1.2.3 VALORES DE LA EMPRESA	10
1.2.4 DATOS DE LA EMPRESA	11
1.2.5 PUESTO O ÁREA DE TRABAJO DEL ESTUDIANTE	12
1.3 PROBLEMAS A RESOLVER	13
1.4.1 GENERAL	13
1.4.2 ESPECIFICOS	13
1.5 JUSTIFICACIÓN	14
CAPITULO II: MARCO TEORICO	15
2.1 BASES TEORICAS	15
2.1.1 ¿Qué es una aplicación móvil?	16
2.1.2 ¿Qué es React native?	19
2.1.3 ¿Qué es JavaScript?	22
2.1.4 ¿Qué es Zeplin?	23
2.1.5 ¿Qué es una API?	24
2.1.6 ¿Qué es Hoppscotch?	27
2.1.7 ¿Qué es Git?	29
2.1.8 ¿Qué es GitHub?	29
2.1.9 ¿Qué es GitKraken?	29
2.1.10 ¿Qué es visual studio code?	30
2.1.11 ¿Qué es un emulador android?	31
2.1.12 ¿Qué es expo go?	32
CAPITULO III: DESARROLLO (PROCEDIMIENTO Y DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS)	33
3.1 Onboarding a NativisProjex	33
3.2 Entorno de desarrollo	34
3.3 Sprint	38
3.3.1 Sprint 1: Maquetación de pantallas “Followers y Following”, “My Publications”, “Payment methods”	38

3.3.2 Sprint 2: Creación del módulo “Privacy Policy”, Conexión front-end con back-end del módulo de “Community”	44
3.3.3 Sprint 3: Continuación de la conexión del módulo de community con el back-end y creación del módulo “Reservations”	49
3.3.4 Sprint 4: Conexión front-back del módulo “Reservations”, Realización de cambios pedidos por el cliente.	52
3.3.5 Sprint 5: Agregar las conexiones del back en la pantalla de "My publications", “Social interests” y en los perfiles de los “Followers y Following”	54
3.3.6 Sprint 6: Cambios solicitados por el cliente, mejora de usabilidad en la pantalla de “Crear publicaciones”	56
3.3.7 Sprint 7: Crear reservas de las propiedades desde los perfiles del módulo de “Community” y el módulo de “Followers y Following”, entrar al perfil de los usuarios seleccionando su nombre desde la lista de publicaciones.....	58
CAPITULO IV: RESULTADOS	61
4.1 RESULTADOS	61
4.1.1 “Followers y Following”, “My Publications”, “Payment methods”	61
4.1.2 Módulo de “Privacy Policy”, Módulo de “Community”	65
4.1.3 Módulo de “Reservations”	70
CAPITULO V: CONCLUSIONES.....	72
5.1 CONCLUSIONES DE PROYECTO, RECOMENDACIONES Y EXPERIENCIA PERSONAL PROFESIONAL ADQUIRIDA	72
5.1.1 CONCLUSIONES DE PROYECTO	72
5.1.2 EXPERIENCIA PERSONAL / PROFESIONAL ADQUIRIDA	73
CAPITULO VI: COMPETENCIAS DESARROLLADAS	74
6.1 COMPETENCIAS DESARROLLADAS Y/O APLICADAS.....	74
CAPITULO VII: FUENTES DE INFORMACIÓN	75
7.1 FUENTES DE INFORMACIÓN.....	75
ANEXOS.....	78
CARTA DE NO INCONVENIENCIA.....	79
Asunto: Carta de No inconveniencia.....	79
Alluxi, S.A. de C.V.....	79
CARTA DE LIBERACIÓN DE LA EMPRESA	80

Tabla de ilustraciones

Ilustración 1 Ubicación de la empresa en Google Maps	11
Ilustración 2 Lenguajes de programación más utilizados en 2022	21
Ilustración 3 Funcionamiento de una API	25
Ilustración 4 Tienda de aplicaciones de Linux (Ubuntu)	34
Ilustración 5 Editor de código Visual Studio Code	34
Ilustración 6 Interfaz de GitKraken para clonar proyectos	35
Ilustración 7 Ejemplo de cómo se visualizan las ramas en GitKraken	36
Ilustración 8 Interfaz completa de GitKraken donde se aprecian las ramas y sus commits	36
Ilustración 9 Visualización de app en emulador Android y iOS	37
Ilustración 10 Visualización de app con Expo Go	37
Ilustración 11 Interfaz de Hoppscotch donde se aprecia el endpoint que se utiliza para obtener información.....	38
Ilustración 12 Maquetación de pantallas de "Followers y Following" usando zeplin	39
Ilustración 13 Endpoint de Follow en Hoppscotch	40
Ilustración 14 Maquetación de la pantalla "My publications" en zeplin	41
Ilustración 15 Maquetación de pantallas de "Payment methods" en zeplin	43
Ilustración 16 Maquetaciones de pantallas de la sección "Community" en zeplin	46
Ilustración 17 Endpoint de Posts en Hoppscotch.....	48
Ilustración 18 Interfaz del endpoint Posts para ver la manera en la que se envían y reciben los datos	48
Ilustración 19 Maquetaciones de las pantallas "Reservations" en zeplin	50
Ilustración 20 Endpoint de Reservations en Hoppscotch	52
Ilustración 21 Reutilización de una sección del endpoint de Posts en Hoppscotch.....	54
Ilustración 22 Maquetación de la pantalla "Social Interests" en zeplin.....	55
Ilustración 23 Endpoint de Social Interests en Hoppscotch.....	56
Ilustración 24 Interfaz del endpoint a utilizar	56
Ilustración 25 Cambios solicitados por el cliente en la sección de Posts (Antes y Después)	57
Ilustración 26 Cambios solicitados por el cliente en la sección de crear un post	57
Ilustración 27 Maquetación de las pantallas de "Create Reservations" en zeplin	59
Ilustración 28 Pantallas funcionales de la sección de "Followers y Following"	61
Ilustración 29 Pantallas funcionales de los perfiles desde "Followers y Following"	62
Ilustración 30 Pantallas funcionales de crear reservaciones desde los perfiles	62
Ilustración 31 Pantallas funcionales de la sección "My publications"	63
Ilustración 32 Pantallas funcionales de la sección "Payment methods"	64
Ilustración 33 Pantallas funcionales de la sección "Privacy Policy"	65
Ilustración 34 Pantallas funcionales de la sección "Community" apartado de posts.....	66
Ilustración 35 Pantallas funcionales de la sección "Community" apartado de búsqueda	67
Ilustración 36 Pantallas funcionales de la sección "Community" apartado de perfiles y crear reservaciones.....	68
Ilustración 37 Pantallas funcionales de la sección "Community" apartado crear publicación.....	69
Ilustración 38 Pantallas funcionales de la sección "Reservations".....	70
Ilustración 39 Pantallas funcionales de la sección "Social Interests".....	71

CAPITULO I: GENERALIDADES DEL PROYECTO

1.1 INTRODUCCIÓN

En un mundo en constante búsqueda de soluciones sostenibles y conscientes con el medio ambiente, cada vez más personas se suman a la tendencia de vivir en casas ecológicas. Estas viviendas no solo contribuyen a reducir el impacto ambiental, sino que también ofrecen un estilo de vida más saludable y económicamente viable.

Conscientes de esta creciente demanda, nos complace presentar esta nueva aplicación de venta y renta de casas ecológicas, desarrollada utilizando la tecnología React Native. Esta innovadora plataforma combina la comodidad y accesibilidad de una aplicación móvil con la conciencia ambiental y las ventajas de vivir en una casa ecológica.

Gracias a React Native, esta aplicación ofrece una experiencia de usuario fluida y dinámica en dispositivos iOS y Android, asegurando que nuestros usuarios puedan explorar una amplia selección de casas ecológicas para la venta o renta de manera intuitiva y conveniente.

Además, la aplicación aprovecha al máximo las capacidades de React Native para brindar a los usuarios información detallada sobre cada propiedad, incluyendo características ecológicas como sistemas de energía renovable, aislamiento eficiente, sistemas de recolección de agua de lluvia y mucho más. También ofrecemos imágenes de alta calidad que permiten a los usuarios explorar cada rincón de las casas ecológicas antes de tomar una decisión.

Ya sea que estés buscando tu hogar ideal o una oportunidad de inversión en propiedades ecológicas, esta aplicación es tu aliada en el proceso. Estamos comprometidos en ofrecer una selección cuidadosamente curada de casas ecológicas que cumplan con los más altos estándares de calidad y sostenibilidad.

1.2 DESCRIPCIÓN DE LA EMPRESA U ORGANIZACIÓN Y DEL PUESTO O ÁREA DEL TRABAJO DEL ESTUDIANTE

1.2.1 DESCRIPCIÓN DE LA EMPRESA

Productos digitales para personas reales.

Construimos apps para startups, PyMEs y emprendedores. Fusionando diseño y tecnología de manera eficiente, creando apps de clase mundial.

1.2.2 MISIÓN DE LA EMPRESA

No queremos hacer solamente lo que nos pidan. Queremos ir más allá e involucrarnos en tu proceso de desarrollo. Queremos conocer la misión detrás tu negocio entender por qué va a ese lugar y ayudarte a llegar a él. Convertirnos en un aliado tecnológico de tu negocio. Para que tú puedas enfocarte en los aspectos más importantes.

Nuestro enfoque nos permite construir software con un futuro en mente; a la vez que construimos relaciones duraderas con nuestros clientes. Creemos que el éxito de un proyecto no se alcanza al final de este, si no que se repite constantemente a lo largo de su evolución.

1.2.3 VALORES DE LA EMPRESA

Pasión: Amamos lo que hacemos y creemos en el aprendizaje constante

Trabajo en equipo: Ponemos en conjunto nuestras habilidades para lograr grandes resultados en un ambiente colaborativo, de reconocimiento y respeto.

Responsabilidad: Siempre entregamos los objetivos que se plantearon desde el inicio y aceptamos los desafíos que éstos conlleven.

Dueños de resultados: Estamos orgullosos del trabajo que realizamos. Nuestra reputación es nuestra carta de presentación.

Aliados: No somos un proveedor de software más. Queremos ser un verdadero aliado de tu proyecto y verlo triunfar.

1.2.4 DATOS DE LA EMPRESA

Nombre de la empresa: ALLUXI S.A DE C.V

Sector: Servicios

Domicilio: América 327 sur, Col. Centro, CP. 64000

Ciudad: Monterrey N.L

RFC: ALU17120642A

Teléfono: 811-500-1600

Titular de la empresa: Ing. Gustavo Alberto Reyes Quiroga – CEO fundador

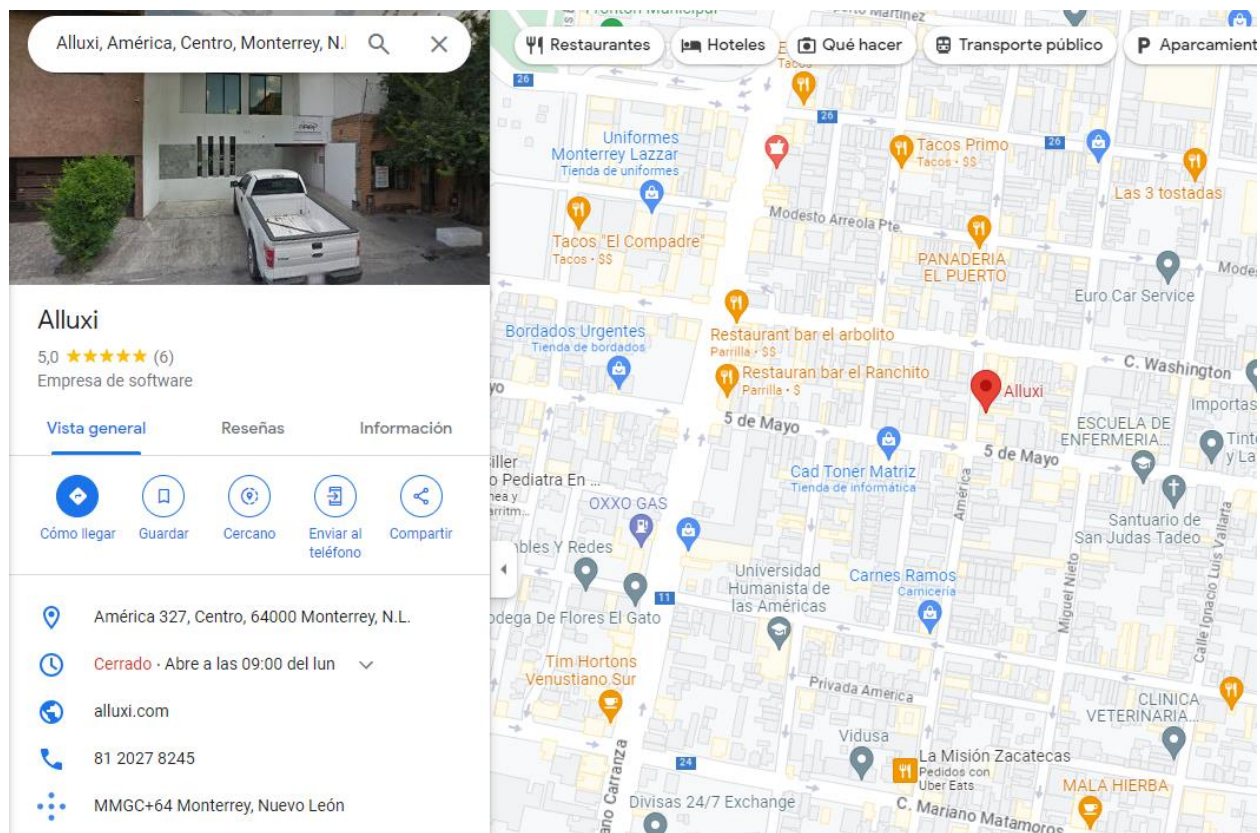


Ilustración 1 Ubicación de la empresa en Google Maps

1.2.5 PUESTO O ÁREA DE TRABAJO DEL ESTUDIANTE

Desarrollador de software encargado de diseñar, crear y elaborar software mediante el uso de lenguajes de programación.

El proceso de trabajo del desarrollador implica crear nuevos sistemas pensando en las necesidades que tiene el usuario y haciendo revisiones y monitorización para controlar que todo el proyecto funcione adecuadamente.

Para este puesto de desarrollador se enfocó más en desempeñarse como desarrollador front-end, responsabilidades a cargo de este puesto:

- Reuniones con el equipo designado a la aplicación
- Reuniones con el cliente y demás equipo para mostrar avances, dudas, comentarios y cambios
- Comunicación con todos los miembros del equipo para que la aplicación funcione correctamente, de igual manera para obtener y dar ayuda
- Traducir el diseño aprobado del cliente a código para poder apreciarlo en la aplicación
- Corrección de errores dentro de la aplicación
- Proactividad al momento de notar mejoras en el transcurso del desarrollo de la aplicación y de esta manera mejorar la experiencia del usuario

1.3 PROBLEMAS A RESOLVER

Nativis solicitó nuestros servicios para construir la aplicación móvil Nativix Projex, una aplicación única para la gestión de alojamientos orientada a alojamientos con necesidades específicas. La app móvil debía ser capaz de ofrecer servicios de alquiler, así como facilitar la compra de inmuebles. Para nuestro cliente era muy importante destacar el espíritu ecológico de este proyecto. Favorecer la comunicación a través de interacciones sociales entre usuarios.

1.4 OBJETIVOS

1.4.1 GENERAL

Desarrollar una aplicación de venta y renta de casas ecológicas usando tecnología React Native y de esta manera fomentar el estilo de vida sostenible, brindando a los usuarios la oportunidad de vivir o rentar hogares respetuosos con el medio ambiente.

1.4.2 ESPECIFICOS

- Diseñar una interfaz intuitiva y atractiva para la aplicación de venta y renta de casas ecológicas.
- Desarrollar funcionalidades que permitan a los usuarios buscar y filtrar las casas ecológicas disponibles según sus necesidades y preferencias.
- Implementar un sistema de registro y autenticación seguro para los usuarios.
- Integrar un sistema de pagos en línea que permita a los usuarios comprar o rentar las casas ecológicas de manera rápida y segura.
- Implementar un sistema de notificaciones para mantener a los usuarios informados sobre nuevas propiedades disponibles o cambios en las propiedades existentes.
- Ofrecer un espacio para que los usuarios puedan dejar reseñas y calificaciones de las casas ecológicas, con el fin de brindar información útil a otros usuarios interesados en vivir o rentar hogares sostenibles.
- Realizar pruebas exhaustivas para garantizar el correcto funcionamiento de la aplicación y brindar una experiencia fluida y sin errores a los usuarios.
- Mantener un registro de las ventas y rentas realizadas a través de la aplicación, con el fin de medir el impacto que se está generando en la adopción de hogares ecológicos

1.5 JUSTIFICACIÓN

Existe una creciente conciencia ambiental y una preocupación por el impacto que nuestras acciones tienen en el planeta. Cada vez más personas están interesadas en adquirir o alquilar viviendas que sean respetuosas con el medio ambiente, lo que representa una oportunidad de negocio en el mercado inmobiliario.

Al desarrollar una aplicación utilizando tecnología React Native, se puede llegar a una audiencia más amplia, ya que esta tecnología permite el desarrollo de aplicaciones multiplataforma, es decir, compatible con diversos sistemas operativos como iOS y Android. Esto permite llegar a un mayor número de usuarios potenciales, aumentando las posibilidades de éxito en el mercado.

Al promover la venta y renta de casas ecológicas, la aplicación contribuye al fomento de la construcción sostenible y al uso de materiales ecológicos. Esto no solo beneficia al medio ambiente, sino que también ayuda a crear conciencia sobre la importancia de adoptar prácticas constructivas más responsables.

Una aplicación móvil de venta y renta de casas ecológicas brinda a los usuarios una mayor comodidad y facilidad de uso. Los usuarios podrán buscar y explorar diferentes opciones de viviendas ecológicas desde la comodidad de su hogar o en cualquier lugar donde tengan acceso a un dispositivo móvil, evitando así desplazarse a diferentes lugares físicos en busca de opciones.

CAPITULO II: MARCO TEORICO

2.1 BASES TEORICAS

En los últimos años, el uso de dispositivos móviles ha experimentado un crecimiento sin precedentes. Los teléfonos inteligentes y las tabletas se han convertido en una parte integral de la vida diaria de muchas personas, ya sea para comunicarse, acceder a información o disfrutar de diversas formas de entretenimiento.

Esta tendencia ha llevado al desarrollo masivo de aplicaciones móviles. Las aplicaciones son programas diseñados específicamente para usarse en dispositivos móviles, dando muchas funcionalidades y servicios a los usuarios.

Las aplicaciones móviles abarcan una gran variedad de categorías, como redes sociales, juegos, organización personal, educación, productividad y mucho más. Estas aplicaciones han transformado la forma en que interactuamos con la tecnología y han permitido una mayor accesibilidad y conveniencia.

Además, el crecimiento de las tecnologías móviles ha impulsado la innovación en términos de hardware y conectividad. Los dispositivos móviles de hoy en día cuentan con cámaras de alta resolución, pantallas táctiles sensibles y una capacidad de procesamiento considerable. Asimismo, las mejoras en la conectividad móvil, como el 5G, han permitido una conexión rápida y estable en cualquier momento y lugar.

El crecimiento de la tecnología móvil también ha tenido un impacto significativo en los negocios. Muchas empresas han optado por tener una presencia en la plataforma móvil para llegar a una audiencia más amplia y ofrecer servicios y productos de manera más conveniente. El comercio móvil ha ganado popularidad, permitiendo a los usuarios realizar compras en línea desde sus dispositivos móviles.

Las aplicaciones móviles han revolucionado la forma en que las personas interactúan con la tecnología y han facilitado enormemente la vida cotidiana.

Primero, las aplicaciones móviles han proporcionado acceso instantáneo a muchos servicios y recursos en la palma de nuestras manos. Antes, las tareas cotidianas, como pedir comida o reservar un taxi, solían requerir tiempo y esfuerzo. Sin embargo, con las aplicaciones móviles, ahora podemos hacer estas actividades con solo unos pocos toques en la pantalla de nuestros dispositivos.

Además, las aplicaciones móviles han permitido la comunicación instantánea y global en tiempo real. Ahora es posible enviar mensajes, hacer videollamadas y compartir contenido con cualquier persona en cualquier parte del mundo, sin importar la distancia.

Esto ha hecho que las relaciones personales y laborales sean mucho más rápidas y eficientes.

Las aplicaciones móviles también han facilitado el acceso a información y conocimiento. Hoy en día, es posible acceder a noticias, artículos, libros electrónicos e incluso cursos académicos directamente desde nuestro dispositivo móvil. Esto ha democratizado el acceso a la educación y ha abierto nuevas oportunidades de aprendizaje para personas de todas las edades y niveles.

Otro aspecto importante es cómo las aplicaciones móviles han mejorado la productividad y la organización personal. Ahora es posible gestionar el tiempo, realizar seguimiento de tareas y gestionar proyectos desde nuestro dispositivo móvil. Esto ha permitido un mejor control y eficiencia en la vida laboral y personal.

Las aplicaciones móviles han cambiado radicalmente la forma en que las personas interactúan con la tecnología y han facilitado la vida cotidiana al proporcionar acceso instantáneo a servicios, mejorar la comunicación, democratizar el acceso a la información y mejorar la productividad y la organización. Su importancia radica en la capacidad de simplificar y agilizar la vida de las personas, abriendo un mundo de posibilidades y oportunidades.

En resumen, el crecimiento de la tecnología móvil ha llevado a un aumento en el uso de dispositivos móviles y al desarrollo de aplicaciones móviles. Estas tendencias han transformado la forma en que interactuamos con la tecnología, brindando mayor accesibilidad, conveniencia y oportunidades comerciales.

2.1.1 ¿Qué es una aplicación móvil?

Una aplicación móvil, también llamada app móvil, es un tipo de aplicación diseñada para ejecutarse en un dispositivo móvil, que puede ser un teléfono inteligente o una tableta. Incluso si las aplicaciones suelen ser pequeñas unidades de software con funciones limitadas, se las arreglan para proporcionar a los usuarios servicios y experiencias de calidad. (Eduardo, 2022)

Aplicación multiplataforma

Las aplicaciones multiplataforma son apps creadas bajo un único lenguaje de programación que facilita su exportación y su visualización en cualquier dispositivo independientemente de su sistema operativo.

El desarrollo de aplicaciones multiplataforma se puede llevar a cabo de dos maneras:

- Por un lado, usando un lenguaje de desarrollo web como HTML5, CSS o JavaScript. Así se desarrollaría una aplicación como si fuera una web con capacidad para adaptarse a cualquier dispositivo.
- Por otro lado, la creación de aplicaciones móviles multiplataforma también se puede realizar con herramientas de rendering a nativo. En este caso, herramientas como **Flutter o React Native** que son frameworks que generan código nativo para cada sistema operativo. Esto hace que la experiencia de usuario sea igual que una app nativa.

Ventajas de las apps multiplataforma:

- La principal ventaja es que son compatibles con todo tipo de dispositivos y cualquier sistema operativo. Da igual que se visualice en un smartphone, Tablet o pc o que sea para sistemas Android o iOS. Las aplicaciones multiplataforma se adaptan a todo sin necesidad de crear diferentes apps.
- Al ser desarrolladas bajo un mismo lenguaje, el ahorro de tiempo, de costes y de recursos es una realidad. A pesar de esto, su calidad y prestaciones no disminuyen. Las apps multiplataforma tienen un gran rendimiento que no tiene nada que envidiar a las aplicaciones nativas.
- A diferencia de las web apps, las multiplataforma no necesitan usar el navegador ya que se pueden descargar y crear así un acceso directo desde el dispositivo. Además, estas apps tienen una integración completa tanto con el hardware como con el software de cada dispositivo.

Desventajas de las aplicaciones multiplataforma:

- A pesar de que las apps multiplataforma se adaptan a todos los dispositivos, las pruebas y los test no pueden faltar nunca. Por eso, igualmente hay que comprobar que la app se visualiza correctamente. Además, cuando se haga cualquier cambio hay que volver a comprobar que todo funciona bien en el resto de los dispositivos.

- Aunque las aplicaciones sean multiplataformas, hay que generar una **APK (Android) e IPA (iOS)** para que funcione bien, por lo que los SDK para cada plataforma serán necesarios.

En el desarrollo de aplicaciones multiplataforma, se utilizan diversas tecnologías y herramientas para crear aplicaciones móviles que funcionen en diferentes sistemas operativos. Ejemplo de algunos:

Lenguajes de programación:

- **JavaScript:** es uno de los lenguajes más populares en el desarrollo de aplicaciones móviles multiplataforma. Se utiliza junto con otras tecnologías, como HTML y CSS, para crear aplicaciones híbridas.
- **Dart:** es el lenguaje de programación utilizado en el framework Flutter de Google. Flutter permite desarrollar aplicaciones nativas para iOS y Android.

Plataformas de desarrollo:

- **React Native:** es una plataforma de desarrollo de aplicaciones móviles de código abierto desarrollada por Facebook. Permite crear aplicaciones nativas utilizando JavaScript y React.
- **Xamarin:** es una plataforma de desarrollo multiplataforma de Microsoft. Utiliza el lenguaje C# y el framework .NET para crear aplicaciones nativas para iOS, Android y Windows.
- **Flutter:** es un framework de desarrollo de aplicaciones móviles multiplataforma también desarrollado por Google. Utiliza el lenguaje Dart y permite crear aplicaciones nativas para iOS, Android y web.

Frameworks y librerías:

- **Ionic:** es un framework basado en HTML, CSS y JavaScript para crear aplicaciones móviles híbridas. Se puede utilizar junto con Angular para desarrollar aplicaciones con un aspecto y rendimiento nativos.
- **Cordova:** es una plataforma de desarrollo de aplicaciones móviles que utiliza tecnologías web (HTML, CSS y JavaScript) para crear aplicaciones híbridas. Permite acceder a características nativas de los dispositivos mediante plugins.

- **PhoneGap:** es una distribución de Cordova, que ofrece herramientas adicionales para desarrollar aplicaciones móviles multiplataforma de manera más sencilla.

Estas son solo algunas de las tecnologías y herramientas utilizadas en el desarrollo de aplicaciones móviles multiplataforma. La elección de la tecnología depende de las necesidades y preferencias del desarrollador, así como de los requisitos del proyecto.

2.1.2 ¿Qué es React native?

React Native es un marco para Apps de código abierto creado en 2015 por Meta (también conocido anteriormente como Facebook). Con el término código abierto, nos referimos a un código de un programa distribuido de manera libre (e incluso gratuita) y que los usuarios pueden usarlo y modificarlo sin restricciones.

React Native es un framework JavaScript para crear aplicaciones reales nativas para iOS y Android, basado en la librería de JavaScript React para la creación de componentes visuales, cambiando el propósito de los mismos para, en lugar de ser ejecutados en navegador, correr directamente sobre las plataformas móviles nativas, en este caso iOS y Android.

React Native usa el mismo paradigma fundamental de construcción de bloques de UI (componentes visuales con los que interacciona el usuario) que las aplicaciones nativas reales de Android e iOS, pero gestiona la interacción entre los mismos utilizando las capacidades de JavaScript y React.

Características:

- **Compatibilidad Cross-Platform:** ya que la mayoría de las APIs de React Native lo son de por sí, lo cual ayuda a los propios desarrolladores a crear aplicaciones que puede ser ejecutados tanto en iOS como Android simultáneamente con el mismo código base.
- **Funcionalidad nativa:** las aplicaciones creadas mediante React Native funcionan de la misma manera que una aplicación nativa real creada para cada uno de los sistemas usando su lenguaje nativo propio. La unión de React Native junto con JavaScript permite la ejecución de aplicaciones más complejas de manera suave, mejorando incluso el rendimiento de las apps nativas y sin el uso de un WebView.

- **Actualizaciones instantáneas (para desarrollo y/o test):** con la extensión de JavaScript, los desarrolladores tienen la flexibilidad de subir los cambios contenidos en la actualización directamente al dispositivo del usuario sin tener que pasar por las tiendas de aplicaciones propias de cada sistema y sus tediosos ciclos de procesos obligatorios previos. Hay que aclarar que este uno es exclusivo de versiones de desarrollo o para test, es ilegal, y puede llegar a conllevar castigos que llegan hasta la retirada definitiva de la aplicación si se realizan cambios directos sobre código con aplicaciones ya publicados y en producción. La tienda de Apple lleva un control muy exhaustivo sobre este tipo de prácticas.
- **Sencilla curva de aprendizaje:** React Native es extremadamente fácil de leer y sencillo de aprender ya que se basa en los conceptos fundamentales del lenguaje JavaScript, siendo especialmente intuitivo tanto para los ya expertos en dicho lenguaje o incluso para las personas sin experiencia en él, ya que nos provee de un rango muy amplio de componentes, incluyendo ejemplo como los maps y filters típicos que se han usado siempre.
- **Experiencia positiva para el desarrollador:** el propio lenguaje nos motiva y ayuda a la hora de la evolución según aumentamos nuestro conocimiento y dominio del mismo.

Ventajas:

1. **Desarrollo más rápido y barato de las Apps:** React Native ofrece elementos que aceleran considerablemente el proceso de desarrollo, cómo componentes para texto, imagen, animaciones, enlaces etc.

El coste de crear una App híbrida con React Native resulta más barato. Esto es debido a que el desarrollador solo necesita compilar una aplicación una sola vez en JavaScript.

2. **Uso del conocido lenguaje JavaScript:** Según los datos de **SlashData State of the Developer Nation 22nd Edition**, el lenguaje de programación más usado en 155 países es JavaScript, y se estima que hay más de 31 millones de desarrolladores activos en el mundo. Esto se traduce en que la mayoría de los desarrolladores no dudan en escoger React Native para programar ya que está escrito en JavaScript. La ilustración 2 muestra los lenguajes mas populares.

Size of programming language communities in Q1 2022

Active software developers, globally, in millions (n=20,041)

		Most popular in	Least popular in
JavaScript*	17.4 M	Apps for 3rd-party ecosystems, Web	DS/ML, Embedded
Python	15.7 M	DS/ML, IoT apps	Mobile, Web
Java	14.0 M	Cloud, Mobile	DS/ML, Web
C/C++	11.0 M	Embedded, IoT apps	Web, Cloud
C#	10.0 M	Desktop, Games	DS/ML, Mobile
PHP	7.9 M	Web, Cloud	DS/ML, Mobile
Kotlin	5.0 M	Mobile, AR/VR	DS/ML, Desktop
Visual development tools	5.0 M	AR/VR, Desktop	Cloud, Web
Swift	3.5 M	Mobile, AR/VR	Cloud, Embedded
Go	3.3 M	Cloud, Apps for 3rd-party ecosystems	Mobile, DS/ML
Objective C	2.4 M	AR/VR, IoT devices	Desktop, Apps for 3rd-party ecosystems
Rust	2.2 M	AR/VR, IoT apps	Web, Mobile
Ruby	2.1 M	IoT, Apps for 3rd-party ecosystems	Web, Embedded
Dart	1.8 M	Mobile, Apps for 3rd-party ecosystems	Web, Apps for 3rd-party ecosystems
Lua	1.4 M	Games, IoT devices	Mobile, Embedded

/DATA

Ilustración 2 Lenguajes de programación más utilizados en 2022

3. **Código abierto:** React Native permite a los desarrolladores usar sus bibliotecas y marcos de trabajo de forma gratuita y con fácil acceso. Por lo que con una simple descarga ya podrás empezar a programar.
4. **Similar al rendimiento nativo:** React Native desempeña un papel importante durante las mejoras de rendimiento realizadas a través de módulos y controles nativos. La interfaz de usuario que proporciona React Native parece y se siente nativa debido a sus interacciones de JavaScript con el entorno nativo.

5. **Multiplataforma sin problemas:** Los componentes de React envuelven el código nativo existente e interactúan con las API nativas a través del paradigma de UI declarativo de React y JavaScript. Esto permite el desarrollo de Apps nativas y puede permitir que los equipos nativos existentes trabajen mucho más rápidos. El framework puede comunicarse tanto con códigos de JavaScript como con un código nativo, esto es gracias al *Bridge*: un puente que permite la interacción entre ambos códigos. Sin este bridge no hay medios para que el código nativo transmita información al código JavaScript y viceversa.

Desventajas:

1. **Menor rendimiento:** Al no tratarse de Apps nativas, el rendimiento que vamos a tener con las webs responsive y las PWA va a ser menor. React Native es excelente para crear aplicaciones simples con imágenes atractivas. Sin embargo, lograr un rendimiento excelente se convierte en un desafío cuando necesita agregar funciones complejas.
2. **Aplicaciones más pesadas:** Las aplicaciones generadas con React-Native ya de base ocupan más espacio que las aplicaciones generadas con otros sistemas, nativos o híbridos, lo cual implica que pueden tardar más en descargarse, en instalarse y ocupar más espacio en el dispositivo.

(Carrillo, 2022)

2.1.3 ¿Qué es JavaScript?

JavaScript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, te permite crear contenido de actualización dinámica, controlar multimedia, animar imágenes y prácticamente todo lo demás.

JavaScript se clasifica como un lenguaje de guion o interpretado. El código JavaScript se interpreta directamente traducido a código de lenguaje de máquina subyacente con un motor de JavaScript. En el caso de otros lenguajes de programación, un compilador se encarga de compilar todo el código en código de máquina en un paso diferente. En consecuencia, todos los lenguajes de scripts son lenguajes de programación, pero no todos los lenguajes de programación son lenguajes de scripts.

Continuando con el desarrollo de aplicaciones debemos de tener en cuenta el diseño.

El diseño de aplicaciones se centra en la creación de interfaces de usuario intuitivas y atractivas que mejoren la experiencia del usuario al interactuar con una aplicación.

Algunos de los principios fundamentales del diseño de aplicaciones incluyen la usabilidad, la experiencia de usuario (UX) y la interfaz de usuario (UI).

La usabilidad se refiere a la facilidad con la que los usuarios pueden aprender a utilizar una aplicación y realizar las tareas que desean realizar. Una aplicación con una buena usabilidad permite a los usuarios navegar de forma intuitiva, encontrar la información que necesitan y completar acciones sin dificultad. Esto implica tener una estructura de navegación clara, botones y elementos interactivos fácilmente identificables y procesos simplificados.

La experiencia de usuario (UX) se enfoca en cómo se siente y percibe el usuario al interactuar con una aplicación. Dicha experiencia debe ser satisfactoria, agradable y eficiente, garantizando que los usuarios obtengan valor y utilidad de la aplicación. Para lograr una buena UX, es necesario comprender las necesidades y expectativas del usuario, considerar la estética y el diseño visual, y optimizar el rendimiento y la velocidad de la aplicación.

La interfaz de usuario (UI) se refiere al diseño visual y la disposición de los elementos en pantalla. Una interfaz atractiva y bien diseñada hace que la aplicación sea más agradable a la vista y fácil de usar, lo que contribuye a una mejor experiencia de usuario. Esto implica elegir colores y tipos de letra adecuados, utilizar espacios en blanco para destacar elementos clave, y tener una disposición lógica y coherente de los elementos.

Al aplicar estos principios fundamentales de diseño de aplicaciones, se puede mejorar la calidad de una aplicación en varios aspectos. Una aplicación con una buena usabilidad permite a los usuarios realizar tareas de manera eficiente y sin frustración, lo que puede aumentar la satisfacción del usuario. Una buena experiencia de usuario crea una impresión positiva en los usuarios, lo que puede fomentar la fidelidad y la recomendación de la aplicación. Una interfaz de usuario bien diseñada mejora la estética y la capacidad de uso de la aplicación, lo que puede aumentar la confianza y el compromiso del usuario.

2.1.4 ¿Qué es Zeplin?

Zeplin es una herramienta colaborativa entre diseñadores y programadores.

CSS automático

La ventaja de esta herramienta es que al diseñador no le hace falta crear documentos con especificaciones, Zeplin las crea automáticamente, y al desarrollador no le hace falta aprender una herramienta de diseño nueva, Zeplin muestra las medidas, colores, márgenes, etc., de cada elemento y escribe el código CSS. También Sass y Less entre otros.

Ventajas:

- **Archivos en la nube:** trabaja siempre sobre la última versión.
- Las imágenes se recortan y se guardan automáticamente en todos los tamaños posibles.
- **Notas sobre cada elemento:** añade comentarios y discute con el resto del equipo qué opción es mejor.
- **Integración con Slack:** notificaciones instantáneas sobre cambios a todos los miembros del equipo.

Otro dato importante a tomar en cuenta para el desarrollo de aplicaciones es la conexión del cliente con el servidor, para esto nos será de ayuda las API.

El uso de API en el desarrollo de aplicaciones es esencial para la reutilización de código, la integración de servicios, el acceso a datos, la interoperabilidad y la mejora de la experiencia del usuario. Permiten ahorrar tiempo, mejorar la funcionalidad y ofrecer una experiencia más completa y personalizada a los usuarios.

2.1.5 ¿Qué es una API?

El término API es una abreviatura de Application Programming Interfaces, que en español significa ***interfaz de programación de aplicaciones***.

Son definiciones y protocolos utilizados para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software mediante un conjunto de reglas.

Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero. Las API le dan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones; y ofrecen oportunidades de innovación, ideal para diseñar herramientas y productos nuevos (o de gestionar los actuales).

Las API son un medio simplificado para conectar su propia infraestructura a través del desarrollo de aplicaciones nativas de la nube, pero también le permiten compartir sus datos con clientes y otros usuarios externos. Las API públicas aportan un valor comercial único porque simplifican y amplían sus conexiones con los partners y, además, pueden rentabilizar sus datos (un ejemplo conocido es la API de Google Maps). La ilustración 3 muestra de manera visual como es que funcionan las API's

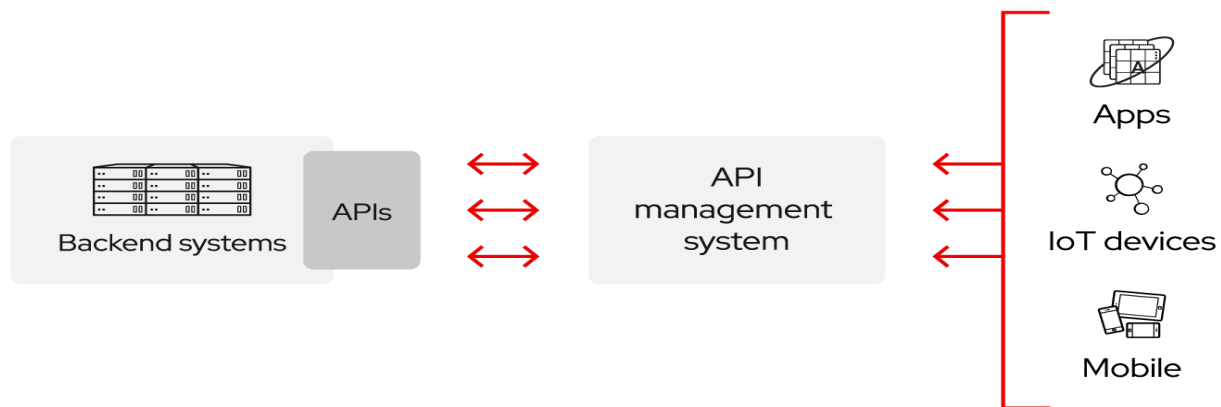


Ilustración 3 Funcionamiento de una API

API de REST

Estas son las API más populares y flexibles que se encuentran en la web actualmente. El cliente envía las solicitudes al servidor como datos. El servidor utiliza esta entrada del cliente para iniciar funciones internas y devuelve los datos de salida al cliente.

¿Qué son las API de REST?

REST significa transferencia de estado representacional. REST define un conjunto de funciones como GET, PUT, DELETE, etc. que los clientes pueden utilizar para acceder a los datos del servidor. Los clientes y los servidores intercambian datos mediante HTTP.

La principal característica de la API de REST es que no tiene estado. La ausencia de estado significa que los servidores no guardan los datos del cliente entre las solicitudes. Las solicitudes de los clientes al servidor son similares a las URL que se escriben en el navegador para visitar un sitio web. La respuesta del servidor son datos simples, sin la típica representación gráfica de una página web.

¿Cómo proteger una API de REST?

Todas las API deben protegerse mediante una autenticación y una supervisión adecuadas. Las dos maneras principales de proteger las API de REST son las siguientes:

1. **Tokens de autenticación:** Se utilizan para autorizar a los usuarios a hacer la llamada a la API. Los tokens de autenticación comprueban que los usuarios son

quienes dicen ser y que tienen los derechos de acceso para esa llamada concreta a la API. Por ejemplo, cuando inicia sesión en el servidor de correo electrónico, el cliente de correo electrónico utiliza tokens de autenticación para un acceso seguro.

2. **Claves de API:** Las claves de API verifican el programa o la aplicación que hace la llamada a la API. Identifican la aplicación y se aseguran de que tiene los derechos de acceso necesarios para hacer la llamada a la API en cuestión. Las claves de API no son tan seguras como los tokens, pero permiten supervisar la API para recopilar datos sobre su uso. Es posible que haya notado una larga cadena de caracteres y números en la URL de su navegador cuando visita diferentes sitios web. Esta cadena es una clave de la API que el sitio web utiliza para hacer llamadas internas a la API.

¿Qué tipos de API hay?

Hay básicamente cuatro tipos de API en lo que se refiere a sus políticas de uso compartido.

API según sus políticas de uso

1. **API's públicas o abiertas:** Las API's públicas también son conocidas como API abiertas y están disponibles para que otros usuarios o desarrolladores las empleen con mínimas restricciones o, en algunos casos incluso, están totalmente accesibles.
2. **API's privadas o internas:** Las API's privadas o internas están ocultas de los usuarios externos y se exponen únicamente para los sistemas internos de una organización. Se emplean para el desarrollo interno de la empresa, optimizando la productividad y la reutilización de servicios.
3. **API's de aliados comerciales:** Las API's de aliados comerciales son aquellas que se exponen entre los miembros de una alianza comercial. Como no están disponibles para todos, se necesita una autorización especial para usarlas.
4. **API's compuestas:** Las API's compuestas utilizan distintos datos o diversas API's de servicio y permiten que los desarrolladores puedan acceder a varios terminales.

API's según sus casos de uso

1. **API de datos:** Las API's de datos proporcionan a varios bancos de datos o proveedores SaaS (Software as a Service o Software como Servicio) acceso CRUD (*Create, Read, Update, Delete*) a conjuntos de datos subyacentes, permitiendo la comunicación entre una aplicación y un sistema de gestión de bases de datos.
2. **API de sistemas operativos:** Este grupo de API's definen cómo las aplicaciones usan los recursos disponibles y servicios del sistema operativo. Por lo que cada OS (*Operative System*) posee un conjunto de API's, por ejemplo, Windows API o Linux API tienen el *kernel-user space API* y *kernel internal API*.
3. **API's remotas:** Este grupo define los estándares de interacción que las aplicaciones tienen en diferentes dispositivos, es decir, un software accede a ciertos recursos ubicados fuera del dispositivo que los solicita, como dice su nombre. Como dos aplicaciones se conectan de forma remota a través de una red, las API's remotas usan protocolos para lograr la conexión.
4. **API's web:** Esta clase de API es la más común, dado que las API's web proporcionan datos que los dispositivos pueden leer y transferirlos entre sistemas basados en la web o arquitectura cliente-servidor.

Los protocolos de API permiten estandarizar el intercambio de datos entre los diferentes servicios web. Esto permite acceder a capacidades en varios sistemas, mediante diferentes lenguajes de programación y en sistemas operativos.

Algunos de los ejemplos de API más conocidas:

- **Google Maps:** gracias a los estándares aplicados por Google, la mayoría de los sitios web pueden usar las API's de Google Maps para integrar mapas.
- **Weather API:** un proveedor de servicios de geolocalización e información meteorológica con diversas API's que van desde el pronóstico del clima, hasta búsquedas de zonas horarias, astronomía y más.

(SYDLE, 2022)

2.1.6 ¿Qué es Hoppscotch?

Hoppscotch es una herramienta para API testing gratuita, de código abierto y muy liviana para crear y utilizar peticiones de forma rápida. Está disponible a través de una web app y también posee una versión PWA (aplicación web progresiva) y si bien no requiere registro para probar el servicio, es conveniente para guardar los resultados en la nube.

Hoppscotch ofrece una interfaz de usuario donde podemos encontrar soporte para Rest, WebSocket y GraphQL. Puedes enviar fácilmente solicitudes a las API en diferentes entornos utilizando variables de entorno de Hoppscotch.

Ventajas de Hoppscotch:

- Ligera y rápida.
- Generador de documentación.
- Soporte para WebSocket permite también SSE, Socket.IO y MQTT
- Soporte dedicado a GraphQL permite ver y crear solicitudes fácilmente
- Características en aumento gracias a la colaboración de la comunidad.

Desventajas de Hoppscotch:

- Poca documentación.
- Faltan características para la automatización de pruebas.
- Necesidad de agregar algunas características por CLI.

Para proyectos que requieran pruebas manuales y ágiles, Hoppscotch es la herramienta ideal.

Finalmente, como buen desarrollador tanto front-end y back-end es siempre importante tener en cuenta el uso de un sistema de control de versiones. El control de versiones, también conocido como "control de código fuente", es la práctica de rastrear y gestionar los cambios en el código de software. Los sistemas de control de versiones son herramientas de software que ayudan a los equipos de software a gestionar los cambios en el código fuente a lo largo del tiempo.

El software de control de versiones realiza un seguimiento de todas las modificaciones en el código en un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden ir hacia atrás en el tiempo y comparar las versiones anteriores del código para ayudar a resolver el error, al tiempo que se minimizan las interrupciones para todos los miembros del equipo.

Los desarrolladores de software que trabajan en equipos están escribiendo continuamente nuevo código fuente y cambiando el que ya existe. El código de un proyecto, una aplicación o un componente de software normalmente se organiza en una estructura de carpetas o "árbol de archivos". El control de versiones ayuda a los equipos a resolver este tipo de problemas al realizar un seguimiento de todos los cambios individuales de cada colaborador y al contribuir a evitar que el trabajo concurrente entre en conflicto.

Los equipos de software que no usan ningún control de versiones suelen tener problemas como no saber qué cambios están disponibles para los usuarios o crear cambios

incompatibles entre dos partes no relacionadas que deben desvincularse y revisarse exhaustivamente. (Perez, 2022)

2.1.7 ¿Qué es Git?

Se trata de un sistema de control de versiones distribuido. Esto significa que cualquier desarrollador del equipo que tenga acceso puede gestionar el código fuente y su historial de cambios utilizando las herramientas de línea de comandos de Git.

A diferencia de los sistemas de control de versiones centralizados, Git ofrece ramas de características. Esto significa que cada ingeniero de software en el equipo puede dividir una rama de características que proporcionará un repositorio local aislado para hacer cambios en el código.

Las ramas de características no afectan a la rama maestra, que es donde se encuentra el código original del proyecto. Una vez que se hayan realizado los cambios y el código actualizado esté listo, la rama de características puede fusionarse de nuevo con la rama maestra, que es la forma en que se harán efectivos los cambios en el proyecto.

Si esos cambios en el código fuente arruinan el proyecto cuando se implementan, GitHub hace que sea fácil revertirlos con unos pocos clics, y se recuperará la versión anterior del proyecto.

2.1.8 ¿Qué es GitHub?

GitHub es un servicio basado en la nube que aloja un sistema de control de versiones (VCS) llamado Git. Éste permite a los desarrolladores colaborar y realizar cambios en proyectos compartidos, a la vez que mantienen un seguimiento detallado de su progreso.

GitHub es una plataforma de gestión y organización de proyectos basada en la nube que incorpora las funciones de control de versiones de Git. Los usuarios de GitHub pueden rastrear y gestionar los cambios en el código fuente en tiempo real, y acceden a las demás funciones de Git disponibles en el mismo lugar.

2.1.9 ¿Qué es GitKraken?

GitKraken es una herramienta multi plataforma (Mac, Windows y Linux) que nos ayuda a manejar Git de manera sencilla, lo cual incide en nuestra productividad.

GitKraken se integra transparentemente con GitHub, Bitbucket y GitLab y soporta a través

de su muy bien lograda interfaz de usuario funcionalidades de arrastrar y soltar para simplificar tareas que pudieren ser complicadas como unir (*merge*), integrar cambios de una rama a otra (*rebase*) y empujar o publicar (*push*).

Entre las características más destacadas de GitKraken están:

- Puedes tener múltiples perfiles.
- Se integra perfectamente con GitHub Enterprise, GitLab (tanto hospedado como no), Bitbucket y VSTS.
- Editor para poder ver ramas, merging, historial de commits.
- Posibilidad de configurar modo oscuro.
- Simplicidad en cuanto a las acciones merge, rebase y push. También admite crear, clonar y añadir repositorios de forma remota, así como ver y crear pull requests.
- Puedes abrir repos, configurar favoritos, organizar tus productos y grupos, etc.
- Incluye editor de código built-in para editar ficheros directamente desde él. Incluso puedes usar diff para ver diferencias en el código, y tiene capacidad para resaltar texto, buscar, etc.
- Por supuesto, puedes usar Gitflow, Git Hooks, LFS y submódulos.

2.1.10 ¿Qué es visual studio code?

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sin número de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

Características

- **Multiplataforma:** Es una característica importante en cualquier aplicación y más si trata de desarrollo. Visual Studio Code está disponible para Windows, GNU/Linux y macOS.
- **IntelliSense:** Esta característica está relacionada con la edición de código, autocompletado y resaltado de sintaxis, lo que permite ser más ágil a la hora de escribir código. Como su nombre lo indica, proporciona sugerencias de código y terminaciones inteligentes en base a los tipos de variables, funciones, etc. Con la ayuda de extensiones se puede personalizar y conseguir un IntelliSense más completo para cualquier lenguaje.

- **Depuración:** Visual Studio Code incluye la función de depuración que ayuda a detectar errores en el código. De esta manera, nos evitamos tener que revisar línea por línea a puro ojo humano para encontrar errores. VS Code también es capaz de detectar pequeños errores de forma automática antes de ejecutar el código o la depuración como tal.
- **Uso del control de versiones:** Visual Studio Code tiene compatibilidad con Git, por lo que puedes revisar diferencias o lo que conocemos con git diff, organizar archivos, realizar commits desde el editor, y hacer push y pull desde cualquier servicio de gestión de código fuente (SMC). Los demás SMC están disponibles mediante extensiones.
- **Extensiones:** Las extensiones nos permiten personalizar y agregar funcionalidad adicional de forma modular y aislada. Por ejemplo, para programar en diferentes lenguajes, agregar nuevos temas al editor, y conectar con otros servicios. Realmente las extensiones nos permiten tener una mejor experiencia, y lo más importante, no afectan en el rendimiento del editor, ya que se ejecutan en procesos independientes.

VS Studio Code incluye una terminal con todas las funciones, la cual se inicia fácilmente en el directorio de trabajo. La terminal integrada puede utilizar cualquier Shell instalado en el equipo, como PowerShell, Bash o cualquier otro. Contar con una terminal en el propio editor es de gran utilidad para ejecutar diferentes comandos necesarios cuando estamos desarrollando.

2.1.11 ¿Qué es un emulador android?

Un emulador es un mecanismo que permite a los usuarios simular dispositivos Android en una computadora para que se pueda probar una app en diferentes dispositivos y niveles de API de Android sin necesidad de contar con los dispositivos físicos.

Existen varios Emuladores en este sistema operativo, uno de los más populares Android Studio en la que los desarrolladores pueden crear y probar sus aplicaciones en Android.

2.1.12 ¿Qué es expo go?

Expo es un framework para desarrollar rápidamente aplicaciones React Native. Es como Laravel o sinfonía de los desarrolladores PHP o Ruby on Rails para los desarrolladores de Ruby. Expo proporciona una capa encima de las API React Native para hacerlos más fáciles de utilizar y administrar. También proporciona herramientas que facilitan el arranque y probar aplicaciones React Native. Por último, proporciona servicios que normalmente sólo están disponibles cuando se instala un componente nativo reaccione de terceros y componentes de interfaz de usuario. Todos los se hacen disponibles a través del SDK de la Expo.

Ventajas

- Sin la necesidad de tener Xcode o Android Studio, puedes correr tu proyecto de Expo desde la app para Android y iOS y poder compartirla con el link especial de Expo o al escanear su QR.
- Mantiene la visión de React Native al conservar y maximizar el poder del uso de JavaScript al desarrollar aplicaciones nativas.
- Te proporciona componentes para la UI que React Native por sí solo no tiene, aunque las apps nativas sí.
- Nos provee acceso a servicios complejos pero indispensables como Push Notifications, Local Storage, etc.
- Al hacerlo en Expo, te evitas la instalación de componentes externos que afectarían el rendimiento de la misma.

Desventajas

- La creación de un archivo en Expo naturalmente es desde el Expo XDE, aunque la comunidad de React ha desarrollado el inicio rápido create-react-native-app que tiene por default el uso de Expo.
- Faltan bastantes funcionalidades nativas por agregar a SDK, aunque con cada actualización va creciendo su catálogo.

CAPITULO III: DESARROLLO (PROCEDIMIENTO Y DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS)

3.1 Onboarding a NativisProjex

Durante el Onboarding se dio la bienvenida al equipo de trabajo, se asignaron canales de comunicación para poder tener contacto con los integrantes de la empresa y del equipo, de esta manera poder compartir los avances (daily), dudas o problemas que no pudieras resolver y poder obtener ayuda.

Se dio una introducción sobre que trataba la aplicación a realizar y así poder tener un conocimiento más amplio para el momento de programar los requerimientos del cliente.

Mediante una pequeña presentación se dio a conocer quién era el Project manager, los desarrolladores back-end y front-end y quien era el QA, de esta manera saber a quién recurrir en caso de tener algún problema al momento de realizar alguna tarea. Continuando con la presentación se explicó la manera en la que se realizaban las tareas y de que herramientas se apoyaban para poder tener un mejor control sobre estas.

Estas tareas se asignaban mediante JIRA, el cual permitía de manera más fácil observar en un tablero las tareas que había, poder cambiarlas de estado (por hacer, en progreso, testing y QA), asignarles un storypoint (tiempo estimado de realización de la tarea), etc.

Se explica que se hacen sesiones cada 2 semanas, se llaman sprint, en las que primero se revisa que se terminó a tiempo, que faltan tareas por realizar, que siguen en proceso, después resueltos estos puntos se pasa a la asignación de nuevas tareas o a la continuación de tareas atrasadas y así poder seguir trabajando, como desarrolladores debemos asignarle un tiempo estimado a las tareas asignadas para que el manager lleve un control de estas y pregunte actualizaciones de tarea cada cierto tiempo. Esto se hace ya que al terminar cierto sprint se tiene junta con el cliente para mostrarle avances y él pueda decirnos si le parecen bien o requiere algunos cambios.

Una vez explicados roles de trabajo y el funcionamiento de cómo se calificaban las tareas se explicó las herramientas utilizadas en la app. Se hicieron pequeñas sesiones individuales con los desarrolladores para poder comenzar a levantar el proyecto de manera local.

3.2 Entorno de desarrollo

Para preparar el entorno de desarrollo para el proyecto, se enfocó en el desarrollo front-end, para lo cual se iba a necesitar lo siguiente:

1. Instalación de algún editor de código, en este caso se utilizó VSC (visual studio code). Para hacer la instalación solo hay que seguir las instrucciones dependiendo del sistema operativo que se tenga, para mi caso se utilizó LINUX. La cual se encontraba desde la tienda de UBUNTU.

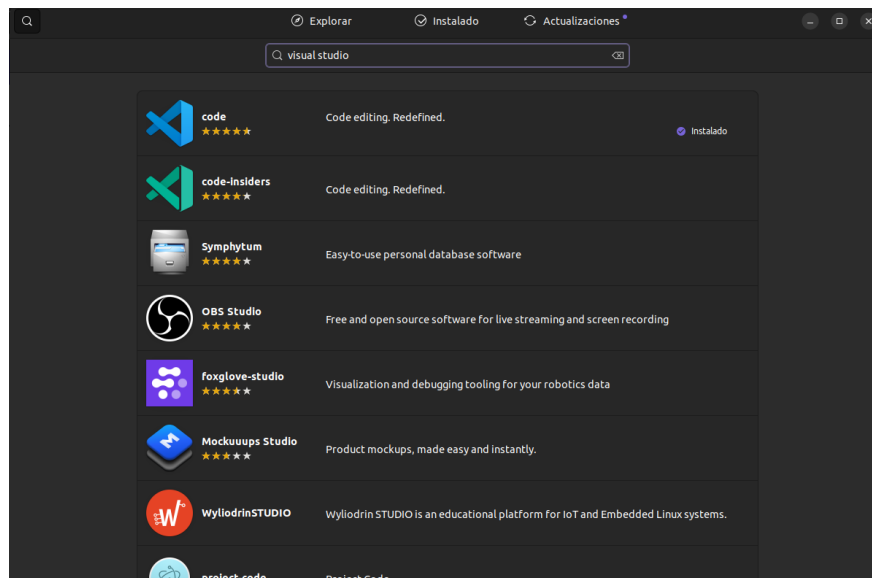


Ilustración 4 Tienda de aplicaciones de Linux (Ubuntu)

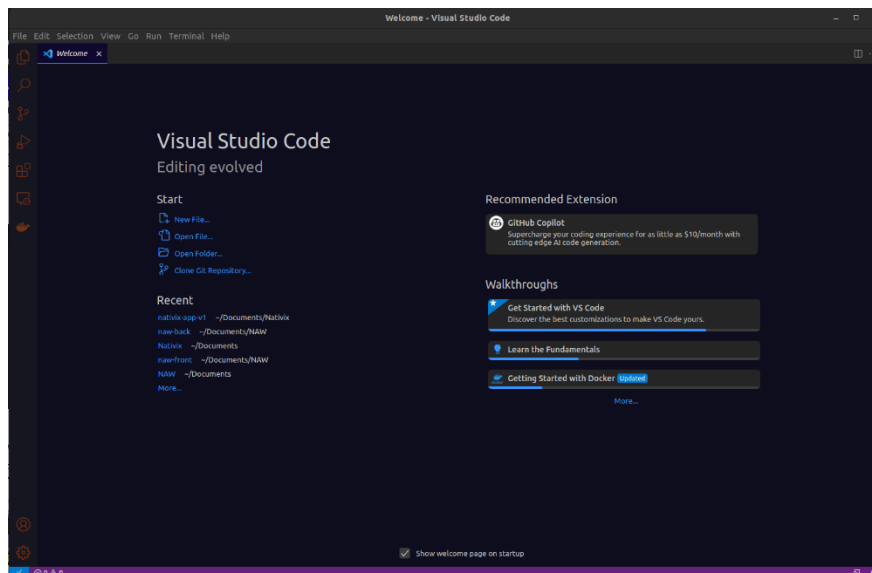


Ilustración 5 Editor de código Visual Studio Code

2. Se asignaron permisos a los repositorios de GitHub para el proyecto de NATIVIS. Tras obtener los permisos se clonó el proyecto. Para este paso fue necesario el uso de GitKraken ya que este nos facilita más este proceso.

Para la instalación de GitKraken se hizo desde la página de <https://www.gitkraken.com/github-student-developer-pack-bundle>, es importante utilizar el mismo correo de GitHub con GitKraken y evitar confusiones al momento de realizar cambios.

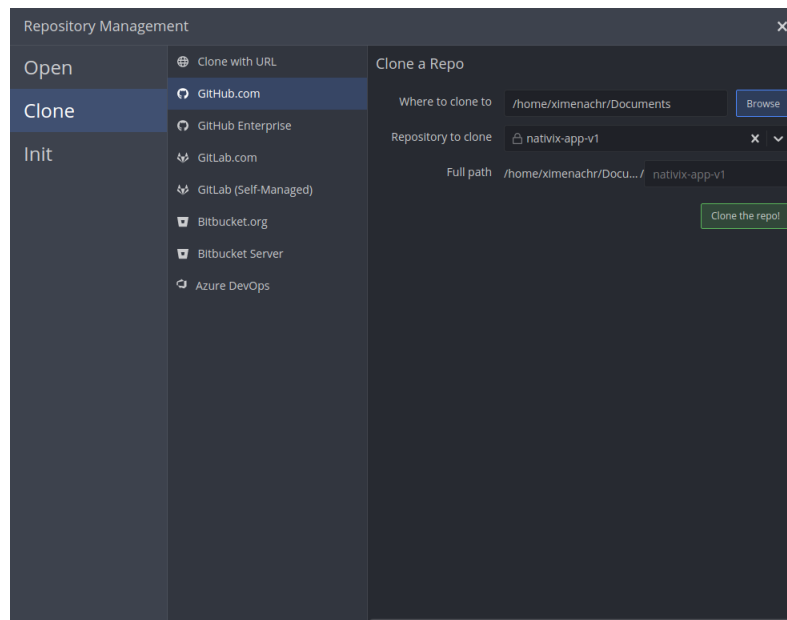


Ilustración 6 Interfaz de GitKraken para clonar proyectos

Con GitKraken nos da varias opciones para poder clonar nuestro proyecto. Ya hecho esto, tendremos el proyecto localmente.

En las reuniones que se tuvieron con los desarrolladores se explicó que la manera de trabajar era:

- Crear ramas desde la rama develop.
- El nombre que deberían de llevar las ramas dependía de lo que se fuera a hacer: tarea nueva (feature/NAT-01_Nombre-de-la-tarea), corrección de algún bug en alguna de las ramas (bugfix/NAT-01_Nombre-de-la-rama-que-tiene-el-bug) o algún cambio mínimo (hotfix/NAT-01_Nombre-de-la-rama-para-agregar-el-cambio).

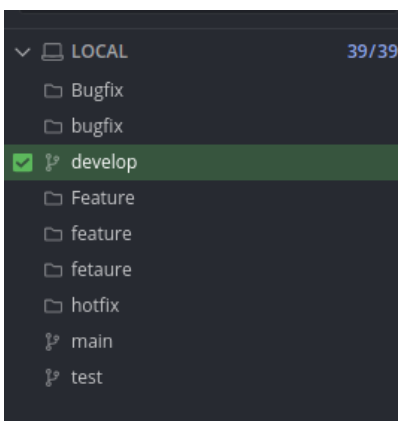


Ilustración 7 Ejemplo de cómo se visualizan las ramas en GitKraken

- Para hacer commits en las ramas deberían de agregarse ciertos comentarios como: ADD, WIP (work in progress), FIX. Todo dependiendo de lo que se estuviera haciendo.
- Asignar a cierta persona al momento de mandar estos cambios para que fueran aprobados y pudieran hacer merge de las ramas con develop.

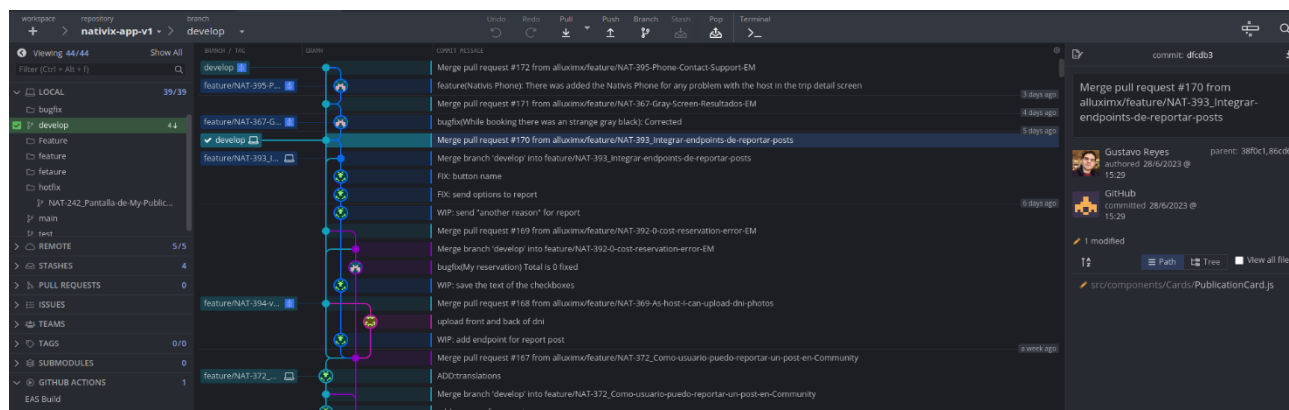


Ilustración 8 Interfaz completa de GitKraken donde se aprecian las ramas y sus commits

3. Para conocer la interfaz del proyecto se ocupa zeplin, en la cual podemos observar el maquetado de la aplicación con los diseños que el cliente ya aprobó anteriormente, en este maquetado se tenían los siguientes módulos ya aprobados:

- Sign up / Login
- Explore
- Search
- Results
- Trips
- Notifications
- Profile
- Community
- Host-Properties
- Host-New property
- Host-Reservations
- Host-Profile

4. Para poder observar la aplicación y los cambios teníamos dos opciones para visualizarlo, utilizando un emulador de Android o usando expo Go. En caso de querer ver estos cambios en iOS teníamos disponible un server de Mac en el cual podíamos hacer uso del emulador de iOS y verificar como se verían nuestros cambios desde ahí

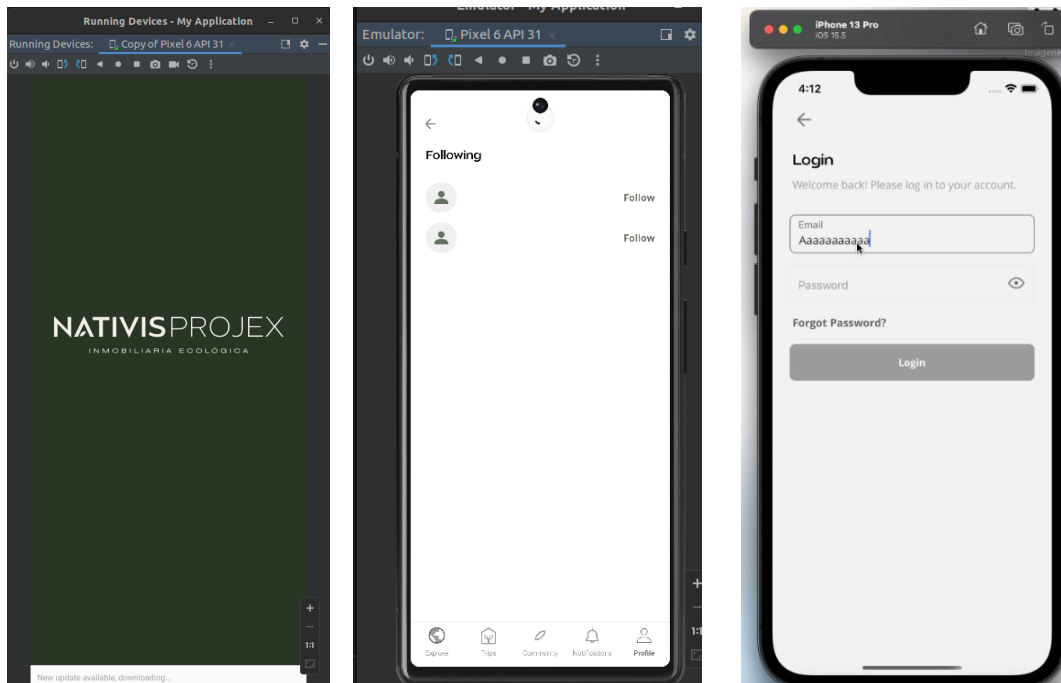


Ilustración 9 Visualización de app en emulador Android y iOS

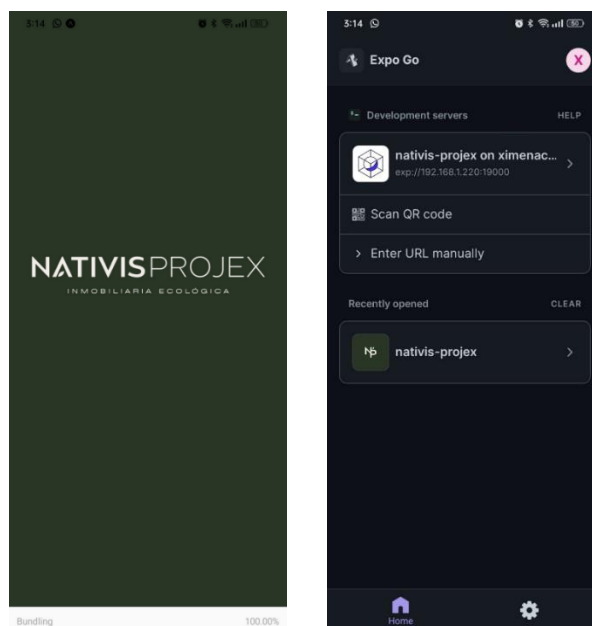


Ilustración 10 Visualización de app con Expo Go

- Se otorgaron permisos para poder usar la herramienta de Hoppscotch, en la cual se tenía la información del back-end, esta información se denominaba como endpoints. Los endpoints nos ayudarían a hacer las conexiones entre front-end y back-end y la herramienta de Hoppscotch nos ayudaría a hacer testing para verificar como deberían de funcionar el envío, la obtención, modificación y eliminación de los datos

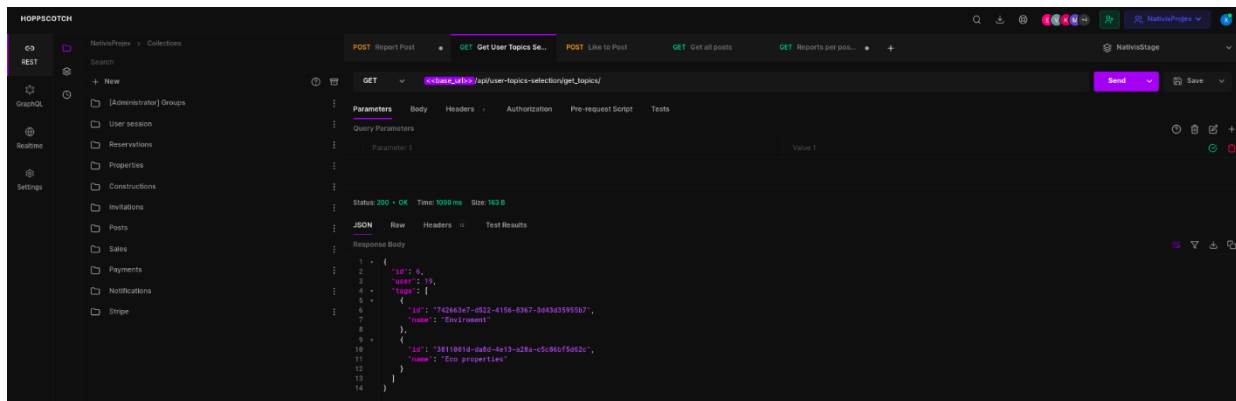


Ilustración 11 Interfaz de Hoppscotch donde se aprecia el endpoint que se utiliza para obtener información

Ahora si teniendo en cuenta toda esta información y todos los permisos necesarios teníamos preparados nuestro entorno de desarrollo el cual nos serviría para comenzar a trabajar en el proyecto.

3.3 Sprint

3.3.1 Sprint 1: Maquetación de pantallas “Followers y Following”, “My Publications”, “Payment methods”

En este primer sprint se me pidió desarrollar el módulo de “Followers y Following”, “My publications” y “Payment methods”.

Durante este sprint el trabajo que se iba a realizar era el maquetado de las pantallas en la aplicación, ya que por el momento solo se tenía el diseño en zeplin. Para poder trabajar estas tareas fue necesario realizar un análisis de los requisitos que se necesitarían para estas pantallas.

3.3.1.1 “Followers y Following”

Se revisó primero el diseño de “Followers y Following”. Este diseño ya lo aprobó antes el cliente.

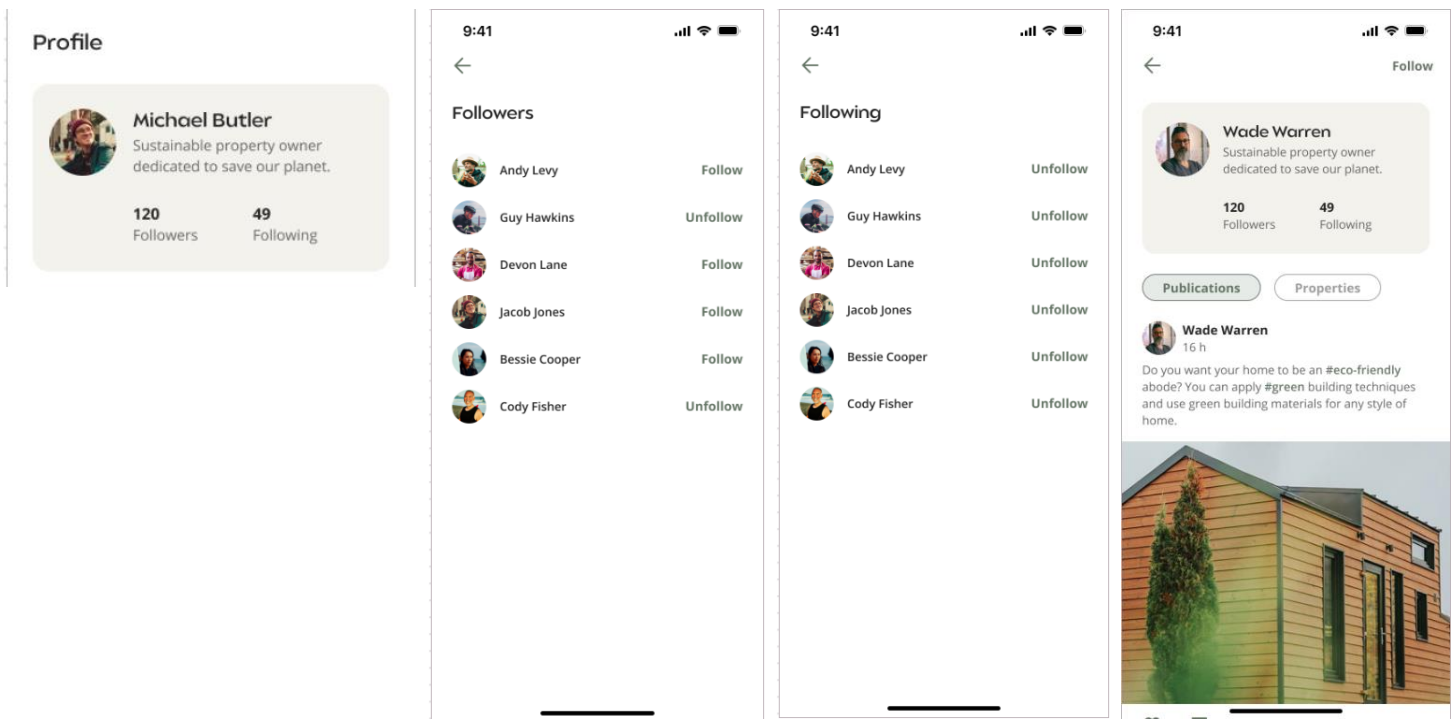


Ilustración 12 Maquetación de pantallas de "Followers y Following" usando zeplin

Basándonos en el diseño se procedió a listar los requerimientos funcionales y no funcionales.

Requerimientos funcionales:

1. El usuario podrá ingresar a las pantallas de Followers y Following desde la sección de perfil
2. El usuario podrá ver a sus followers y following en forma de lista, apreciando la foto y nombre
3. Los usuarios de Followers tendrán un botón de seguir y dejar de seguir (follow/unfollow), en el cual el usuario principal podrá hacer uso de este para actualizar su lista
4. Los usuarios de Following tendrán un botón para dejar de seguirlos, la diferencia en esta pantalla es que todos deberán aparecer con el botón de unfollow activado
5. Si en ninguna de las listas hay usuarios debería de aparecer un mensaje diciendo "no users to display"
6. Si el usuario desea entrar al perfil únicamente tendrá que seleccionar a la persona que desea y este lo mandará a otra pantalla con el perfil

7. La pantalla del perfil seleccionado tendrá que contener su información, descripción y de igual manera sus followers y following. También podrá apreciarse sus publicaciones y propiedades en caso de tener.

Requerimientos no funcionales:

1. La visualización de las listas debe de ser de manera rápida
2. El mensaje debe de aparecer de manera instantánea en caso de que no se cuente con información en estas pantallas
3. La información se envía mediante la API REST para que pueda recibir el token de autenticación del usuario que está solicitándola
4. Las listas deben de poder actualizarse ante cualquier cambio, sin la necesidad de reiniciar la aplicación

Desarrollo:

Se crearon pantallas en blanco, para poder tener listo los lugares a donde navegarían los botones al darles clic, se maquetó el orden de los elementos para poder mostrar a los usuarios, primero aparecería la foto seguido de su nombre y el botón de follow/unfollow, se creó el mensaje que aparecería en caso de no haber usuarios.

Terminada la maquetación se hicieron pruebas en los endpoints para verificar de qué manera funcionaban los datos. Se necesitó ayuda del back para poder modificar el endpoint de follow para que no duplicara datos y este mismo tuviera la función de unfollow.

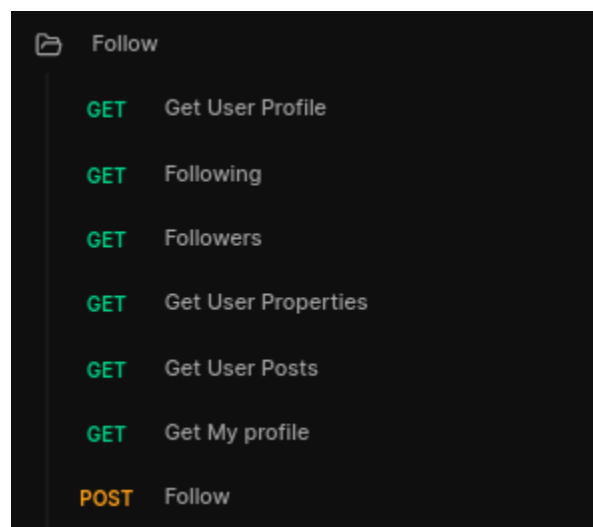


Ilustración 13 Endpoint de Follow en Hoppscotch

Una vez revisada esta parte se realizó la conexión de front y back, para que el usuario pudiera apreciar los datos verdaderos de sus followers y following, de igual manera se agregó la funcionalidad al botón de follow/unfollow para que actualizara los datos y no estuviera solo maquettato. Por fin teniendo estas pantallas se procedió a maquettar la pantalla que nos mostraría el perfil de la persona.

En esta pantalla se agregó un “Profile card”, debajo de esta información aparecerían dos botones para ver las publicaciones y propiedades que el usuario pudiera llegar a tener, en caso de no tener se realizó un mensaje de “No results found”. Esta pantalla solo quedo maquettata, es decir, no contenía datos reales aún.

3.3.1.2 “My publications”

El segundo diseño a implementar en este sprint fue el de My publications.

Primero se revisó el diseño que ya se tenía aprobado en zeplin.



Ilustración 14 Maquetación de la pantalla "My publications" en zeplin

Una vez considerando esto, se listaron los requerimientos para esta pantalla.

Requerimientos funcionales:

1. El usuario podrá entrar desde su perfil en el apartado de configuración a las publicaciones que ha hecho con el botón de “My publications”

2. Al ingresar el usuario podrá ver en forma de lista las publicaciones que este haya hecho
3. En cada publicación podrá ver su nombre, la fecha de publicación, la foto y en la parte inferior de esta el número de likes y de comentarios

Requerimientos no funcionales:

1. La información se envía mediante la API REST para que pueda recibir el token de autenticación del usuario que está solicitándola
2. Las publicaciones se ordenarán de la más nueva a la más vieja
3. Cada que se haga una publicación nueva esta lista se actualizara automáticamente

Desarrollo

Se creo una pantalla en blanco para poder agregarle navegación al botón de My publications y este tuviera a donde dirigirse. Se agregaron los requerimientos necesarios para esta pantalla. Desde el nombre de la persona hasta la información de la publicación. Se maquetaron los botones de like y comentario para que tuvieran el diseño que ya se había aprobado anteriormente.

Para este sprint se dejó maquetada únicamente ya que aún no se tenían claros los endpoints a utilizar.

3.3.1.3 “Payment methods”

El último diseño realizado en este sprint fue el de métodos de pago, para este último fue mucho más fácil realizar la pantalla, porque antes se había agregado está, pero en otra sección diferente.

Se reviso el diseño utilizado en la sección de reservas, ya que prácticamente eran las mismas pantallas.

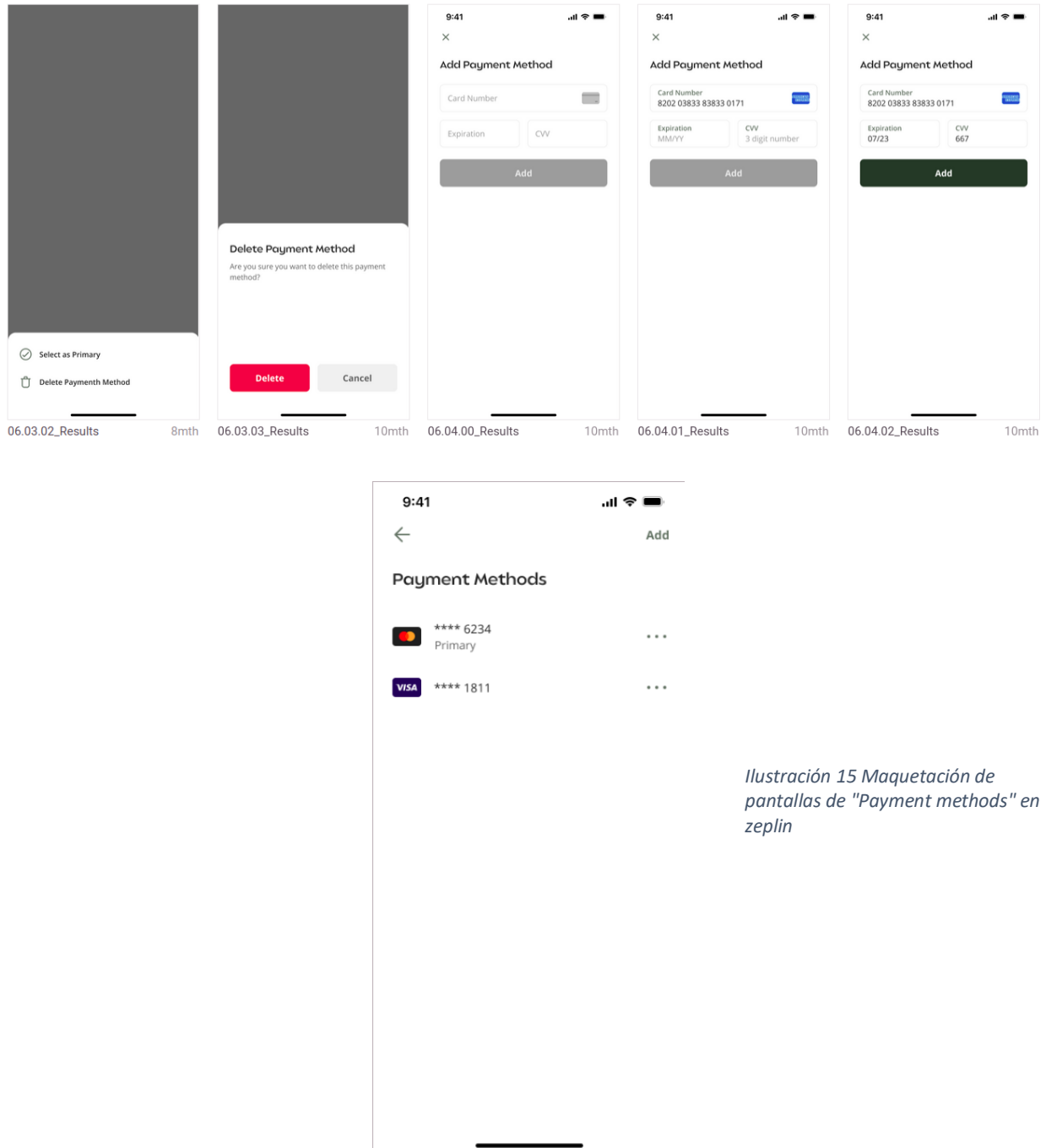


Ilustración 15 Maquetación de pantallas de "Payment methods" en zeplin

Teniendo el diseño listo se listaron los requerimientos necesarios.

Requerimientos funcionales:

1. El usuario podía entrar desde su perfil a los métodos de pago que tuviera registrados. En caso de no tener aparecería un mensaje informando esto.
2. Estos métodos aparecerían en forma de lista con un botón de opciones
3. La opción que el usuario tendría para los métodos de pago seria eliminarlo

4. Se agregaría un botón de add que le indicaría al usuario que ese botón serviría para agregar más métodos
5. Si el usuario quería agregar más, debería de dar clic en el botón de add y este lo redirigiría a otras pantallas y poder hacer el proceso
6. La pantalla de agregar métodos contendría campos vacíos para agregar los datos de la tarjeta
7. Si el usuario quería eliminar le aparecería un mensaje para confirmar esa acción

Requerimientos no funcionales:

1. La información se envía mediante la API REST para que pueda recibir el token de autenticación del usuario que está solicitándola
2. Los datos deben de actualizarse de manera rápida
3. Los datos deben de estar protegidos ya que se consideran datos sensibles

Desarrollo:

Para poder maquetar estas pantallas solo fue necesario reutilizar los componentes anteriormente agregados. Únicamente se quedaron maquetadas tanto en la sección de reserva y en la sección del perfil ya que aún no se contaban con los datos del back.

3.3.2 Sprint 2: Creación del módulo “Privacy Policy”, Conexión front-end con back-end del módulo de “Community”

En el segundo sprint se añadieron más tareas por los nuevos requisitos que el cliente había pedido, en este caso se pidió un apartado para la política de privacidad, se corrigieron algunos problemas de funcionalidad del Sprint pasado y se comenzó a trabajar con un nuevo módulo de la aplicación.

3.3.2.1 Política de privacidad

La primera tarea realizada en este sprint fue el apartado de la política de privacidad. Para esta pantalla no se tenía un diseño asignado en zeplin, entonces lo primero que se hizo fue un análisis de los requerimientos

Requerimientos funcionales:

1. El usuario puede entrar a ver la política de privacidad desde el perfil, se agregaría una sección de información, donde se colocaría este modulo
2. El usuario visualizara la política de privacidad en forma de texto simple y así evitar confusión

3. El usuario podría visualizar el logo de la empresa

Requerimientos no funcionales:

1. Esta información únicamente se iba a localizar en el front
2. El texto debe de adaptarse a las pantallas, agregarle fuentes que se usen en las demás partes del texto de la app

Desarrollo:

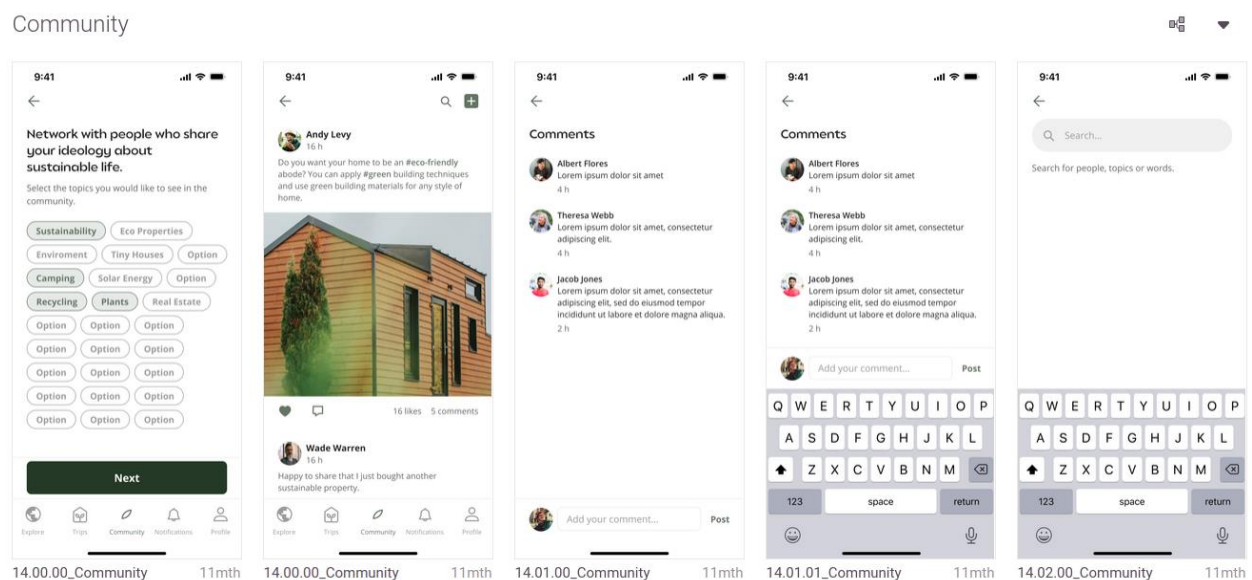
Intenté seguir los colores y las fuentes utilizadas en la aplicación para que no se vieran los cambios, se basó en el documento que el cliente nos envió, que contenía el logo de la empresa seguido de toda la información.

Se agrego un botón en la sección del perfil que nos llevaría a la visualización de esta información. De igual manera se agregó un botón de navegación hacia atrás y en la política de privacidad y poder regresar al perfil.

3.3.2.2 Conexión front-end con back-end del módulo de “Community”

La segunda tarea de este sprint fue comenzar a hacer conexiones de front y back en el módulo de “Community”. Este módulo contenía varias pantallas, todas maquetadas, pero ninguna tenía conexión con los endpoints.

Para comenzar esta tarea primero se verifico todo el diseño asignado en zeplin y así tener un contexto más amplio de que debería de hacer cada cosa.



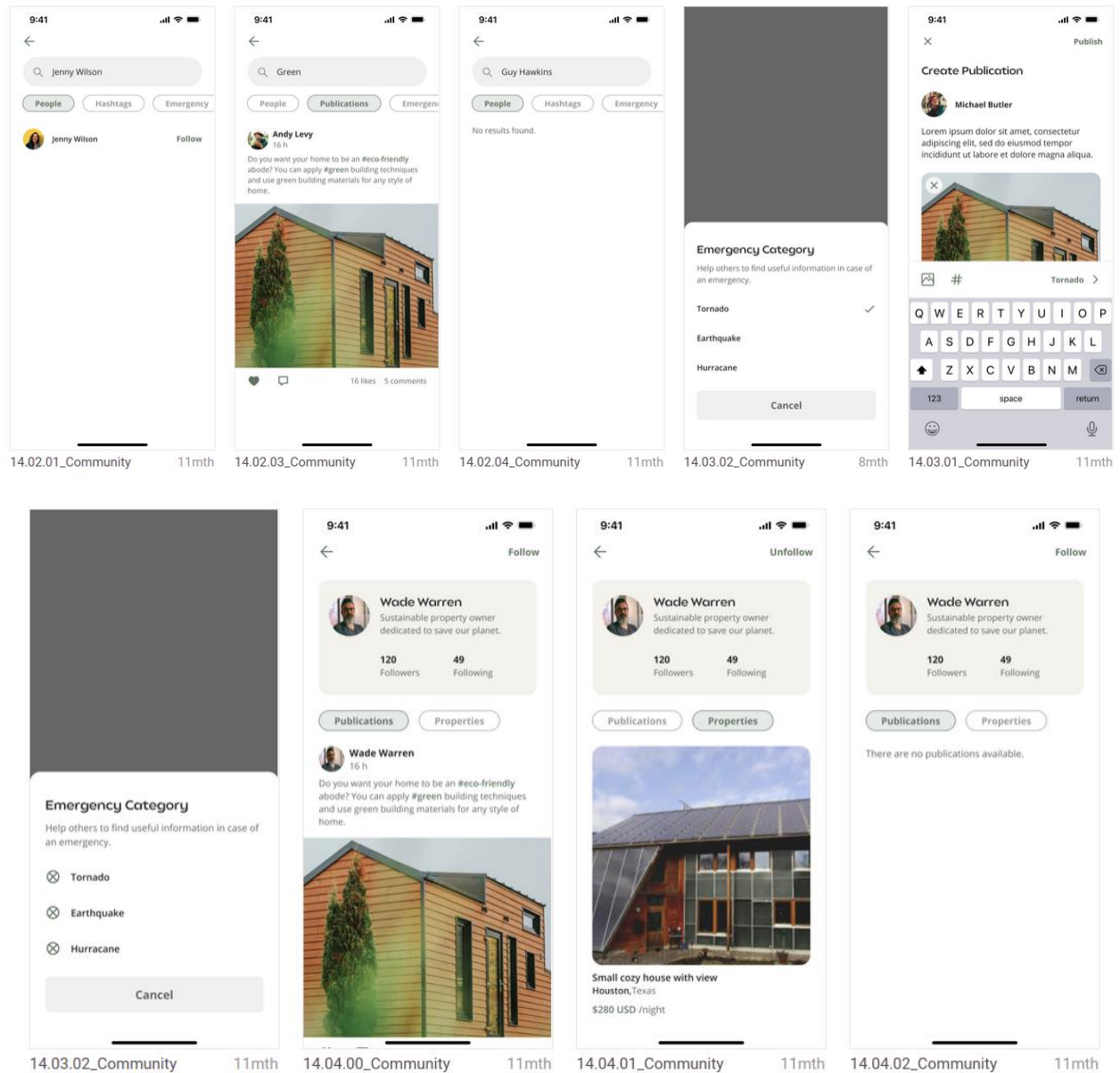


Ilustración 16 Maquetaciones de pantallas de la sección "Community" en zeplin

Requerimientos funcionales:

1. El usuario podría entrar a este módulo desde la barra de botones en la parte inferior de la aplicación
2. Al entrar lo redirigiría a otra pantalla, en la cual mostrara una lista de temas de interés a escoger
3. Seguida de esa pantalla el usuario podrá visualizar publicaciones de acuerdo a los temas que escogió.

4. En esta pantalla, podrá dar like y comentar las publicaciones de igual manera podrá ver los comentarios que se han hecho
5. La sección de los comentarios, aparecen en forma de lista, en orden del más nuevo al más viejo y con fecha de publicación.
6. En la parte superior de la pantalla se ubican botones de navegación, como regresar, buscar y crear una publicación
7. El botón de buscar llevará al usuario a otra pantalla en la cual tendrá un recuadro para ingresar el texto, debajo de esto se tendrán 3 botones más (buscar por personas, por hashtags, por temas de interés)
8. Al buscar una persona el usuario podrá seguirla y esta aparecerá en nuestra lista de following, de igual manera podría entrar al perfil de esta. En la que podríamos ver sus publicaciones y propiedades
9. Al buscar hashtag o por tema nos aparecerían las publicaciones en lista de acuerdo a lo que buscamos
10. El botón que esta para crear publicación te lleva a otra pantalla para comenzar a agregar una nueva publicación. En este apartado, el usuario podrá ver su nombre, un apartado para escribir, en la parte inferior el usuario podrá ver botones para agregar fotos, agregar hashtag, seleccionar temas y otra para seleccionar emergencias naturales.

Requerimientos no funcionales:

1. Hacer la conexión de front-end y back-end mediante el uso de los endpoints asignados para este apartado.
2. La información se envía mediante la API REST para que pueda recibir el token de autenticación del usuario que está solicitándola
3. Toda la información que este en community debe de actualizarse de sin la necesidad de reiniciar la aplicación

Desarrollo:

Después de analizar el diseño que teníamos en zeplin se verifico que este fuera el mismo que ya estaba agregado en la aplicación. Seguido de esto se comenzó a revisar que endpoints se iban a utilizar en este apartado.

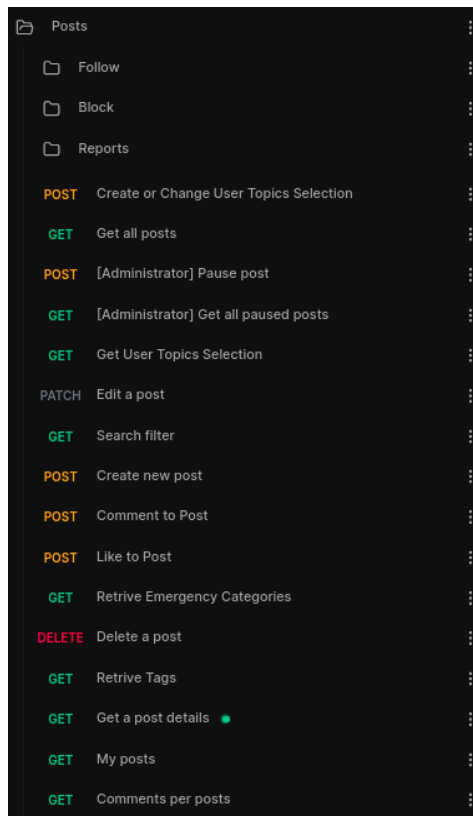


Ilustración 17 Endpoint de Posts en Hoppscotch

Se hicieron pruebas en los endpoints para verificar la manera en la que recibía los datos, los enviaba, etc. Después de las pruebas se agregaron datos con ayuda de hoppscotch para que al momento de empezar con la integración front-back pudiéramos apreciar datos y poder trabajar con ellos.

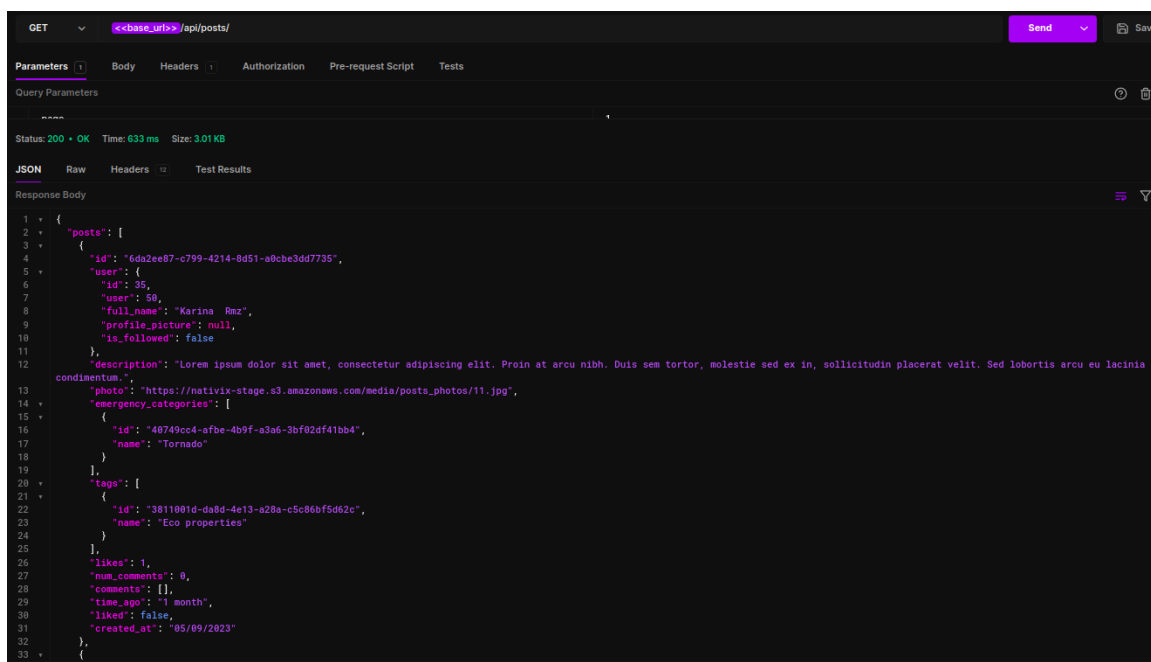


Ilustración 18 Interfaz del endpoint Posts para ver la manera en la que se envían y reciben los datos

Una vez estudiado los endpoints, las funciones que tenía cada uno y haber agregado datos para poder trabajar con ellos se comenzó con el proceso de conexión.

Las conexiones realizadas fueron:

1. Integrar los endpoints que traían la información de los temas de interés a escoger.
2. Integrar el endpoint que traía las publicaciones realizadas, para que se pudieran apreciar de manera real
3. Se agrego funcionalidad a los botones de like, comentarios. Con la conexión al back estos botones ahora reflejaban datos reales como, poner y quitar un like de las publicaciones y este se reflejará en el back, enviar comentarios de manera real y que estos se guardaran en el back.
4. De igual manera se integró el endpoint para poder visualizar los comentarios hechos por otros usuarios.
5. Continuando con los demás botones, se añadió funcionalidad a los botones de búsqueda, así apreciar a los usuarios reales ya registrados anteriormente, buscar publicaciones según las categorías de emergencia que ya estaban en el back y, por último, buscar publicaciones con los hashtags agregados a estas.
6. Se comenzó con integrar el back a la pantalla de crear publicación. Primero se integró el endpoint de la información del usuario para que este pudiera ver su nombre correctamente junto con su foto.

Esta tarea de integración fue necesaria dividirla en dos sprint, las conexiones mencionadas anteriormente fueron las que se lograron terminar en el sprint 2, las conexiones faltantes se realizarían en el Sprint 3.

3.3.3 Sprint 3: Continuación de la conexión del módulo de community con el back-end y creación del módulo “Reservations”

Como ya se mencionó anteriormente faltaban conexiones del módulo de “Community”, esta tarea se le dio un tiempo estimado de terminarla a mediados del sprint 3 y poder realizar otra tarea más, esta segunda tarea era crear otro modulo.

El módulo a crear se llama reservaciones, en el cual el usuario desde su perfil de host pueda ver las reservaciones que tiene sobre sus propiedades.

3.3.3.1 Continuación de la conexión del módulo de “Community” con el back-end

Las siguientes conexiones realizadas fueron:

1. En la pantalla de crear publicación se integró el endpoint que nos mostraría las categorías de emergencias
2. Se agrego funcionalidad al botón de publish para que este enviara los datos al back por medio del endpoint utilizado. Aquí el back tuvo que hacer un cambio ya que al querer la lista de publicaciones no aparecían las que el usuario había hecho.
3. Terminando esta pantalla se continuo la conexión a las pantallas del perfil del usuario. Al entrar al perfil del usuario desde búsqueda era necesario ver toda su información.
4. Se integro el endpoint de la información del usuario buscado, el endpoint de las publicaciones que este había realizado y el endpoint de las propiedades que este tuviera.
5. Igualmente se agregó funcionalidad al botón de follow con el endpoint de seguir a las personas y estas se reflejarán en la lista de seguidos

Con estas conexiones realizadas se finalizó esta tarea, ahora todo funcionaba y mostraba datos verdaderos.

3.3.3.2 Creación del módulo “Reservations”

Para esta segunda tarea, se revisó el diseño asignado en zeplin para después proceder a maquetar en la aplicación

Host - Reservations

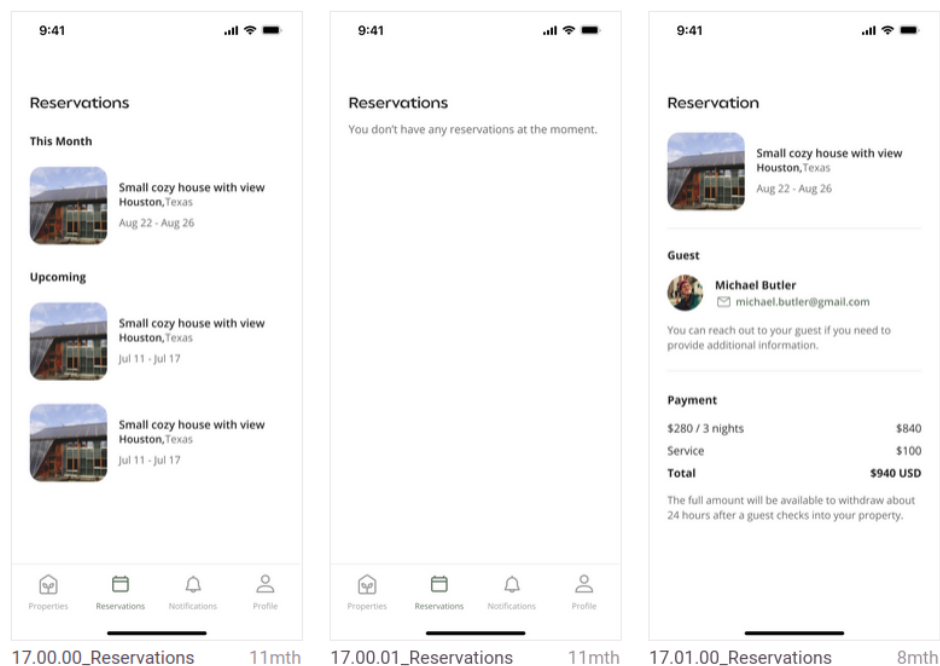


Ilustración 19 Maquetaciones de las pantallas "Reservations" en zeplin

Teniendo en cuenta el diseño se procedió a listar los requerimientos

Requerimientos funcionales:

1. En caso de que el usuario tuviera perfil de host y tuviera propiedades para rentar este podría ver las reservaciones que se hayan hecho
2. Para poder entrar al perfil de host simplemente debería de cambiar de perfil con el botón realizado anteriormente en el perfil
3. Al entrar al perfil de host, la barra inferior cambia de botones de acuerdo para el host
4. Para entrar a las reservaciones hay un botón con el mismo nombre que te lleva a esa pantalla
5. Al entrar a esta pantalla y no tenga reservaciones debe de aparecer un mensaje
6. Si cuenta con reservaciones aparecerá un botón llamado “upcoming” que nos llevará al listado de reservaciones
7. Las reservaciones futuras aparecen en lista de acuerdo a la fecha que tengan, junto con el nombre la propiedad y una foto de esta.
8. Entrando a la reservación el usuario podrá apreciar los datos de la persona que reserva, el tiempo que reservará y el total a pagar, de igual manera la fecha y nombre de la propiedad

Requerimientos no funcionales:

1. Hacer la conexión de front-end y back-end mediante el uso de los endpoints asignados para este apartado.
2. La información se envía mediante la API REST para que pueda recibir el token de autenticación del usuario que está solicitándola
3. La información debería de aparecer de manera inmediata y actualizarse de acuerdo con el back

Desarrollo:

Se realizó primero el maquetado de la aplicación de acuerdo al diseño de zeplin. Se añadieron 3 pantallas, la primera era mostrar un mensaje si no tenía reservaciones o mostrar un botón para llevarnos a la pantalla del listado de las reservaciones. La segunda pantalla aparecía con todas las reservaciones, acomodadas por fechas, se mostraba una pequeña imagen de la propiedad, el nombre de esta, ubicación y fecha de reservación, la tercera pantalla nos llevaba a la descripción de la reservación, mostrando el nombre, lugar, fecha y datos de la persona que reservaba, así como las noches y el precio a pagar.

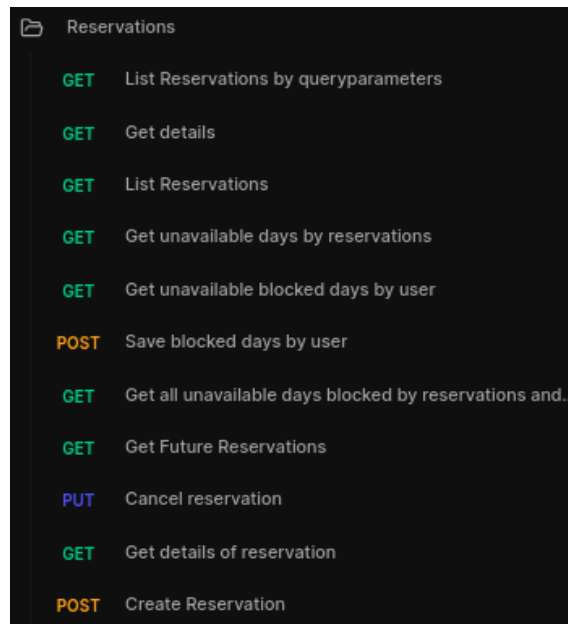


Ilustración 20 Endpoint de Reservations en Hoppscotch

Esta tarea igual fue dividida en dos sprint, en esta primera parte se logró terminar con el maquetado, dejando la conexión del front-back para el sprint 4

3.3.4 Sprint 4: Conexión front-back del módulo “Reservations”, Realización de cambios pedidos por el cliente.

En este sprint se continuó realizando la tarea de creación del módulo de “Reservations”, la cual se estimó terminar a mediados del sprint para poder continuar con algunos cambios que solicito en pantallas trabajadas anteriormente.

3.3.4.1 Continuación de la conexión del módulo de “Reservations” con el back-end

Terminando con el maquetado de estas pantallas de reservaciones era necesario hacer reservaciones desde los endpoint con hoppscotch para poder tener datos y trabajar con ellos. Las conexiones realizadas fueron las siguientes

1. Agregar endpoint del listado de las reservaciones futuras
2. Una vez teniendo este listado, integrar el endpoint para tener los detalles verdaderos de estas
3. Integrar los endpoint de los datos del usuario que reservaba

4. Integrar los endpoint que hacían las cuentas de las noches que se iban a pagar, junto con el endpoint de los precios de cada propiedad.

Estos endpoints tuvieron cambios en el back, los cuales se le pidieron al desarrollador back-end, de esta manera tener un código más limpio desde front.

3.3.4.2 Realización de cambios pedidos por el cliente

Los cambios solicitados por el cliente fueron los siguientes:

1. La pantalla de elegir temas de interés en el módulo de community solo debería de aparecer una vez, ya que el usuario después lo podría cambiar en la sección de su perfil.
2. En la pantalla de crear publicación se agregó otro botón el cual contendría los temas de interés, para que el usuario pudiera agregarlos a sus publicaciones
3. La selección de temas de interés y de emergencia no tendría que ser obligatoria para poder realizar una publicación.

Para el primer cambio solicitado no se necesitó la ayuda de los endpoints, únicamente fue necesario agregar variables que guardaran los datos y con ayuda de condicionales mostrar la pantalla si no había temas seleccionados y si el usuario ya había seleccionado y se tenían guardados datos, los mandaran directamente a las publicaciones

Para el segundo cambio únicamente fue necesario agregar otro botón en la parte inferior de la pantalla y reutilizar el endpoint que se tenía para mostrar los temas de interés, de esta manera el usuario podría apreciar estos topics y agregarlos a sus publicaciones y así las personas que tuvieran esos temas en común pudieran ver sus publicaciones en la sección de community.

Para el último cambio se hicieron pruebas en el endpoint de crear nuevos posts y se llegó a la conclusión que este cambio tendría que hacerse desde back, así que se le pidió al desarrollador back que eliminara esa opción y pudieran hacerse publicaciones sin la necesidad de seleccionarlos.

3.3.5 Sprint 5: Agregar las conexiones del back en la pantalla de "My publications", "Social interests" y en los perfiles de los "Followers y Following"

Anteriormente se había realizado la maquetación de la pantalla de "My publications" y de los perfiles de los usuarios de "Followers y Following" la cual para este sprint era necesaria ya la integración de los endpoints.

Al igual que la pantalla de seleccionar temas de interés en community se tendría que agregar la conexión de front-back en la pantalla de "Social insterests" desde perfil.

3.3.5.1 Agregar las conexiones del back en la pantalla de "My publications"

Para esta tarea únicamente fue necesario realizar la conexión del back, anteriormente ya habíamos hecho algo parecido en el módulo de "Community", esta vez en lugar de traer todas las publicaciones solo vería las mías.

Se reviso el endpoint necesario para esta integración y se procedió a hacer la conexión.



Ilustración 21 Reutilización de una sección del endpoint de Posts en Hoppscotch

Una vez teniendo hecho esto ya podíamos apreciar todas las publicaciones que habíamos realizado en nuestro perfil y no solamente en la lista de publicaciones de community.

3.3.5.2 Agregar las conexiones del back en la pantalla de "Social interests"

Para esta tarea de igual manera ya se había realizado algo parecido en community, la diferencia seria que esta pantalla siempre aparecería y podrían cambiarse los temas cada que el usuario quisiera.

Primero se revisó el diseño en zeplin y se analizaron los requerimientos necesarios.



Ilustración 22 Maquetación de la pantalla "Social Interests" en zeplin

Requerimientos funcionales:

1. Se cambiaría la información de la pantalla, pero serían los mismos temas
2. El usuario puede cambiar los temas que selecciono en community o agregar mas
3. Los temas aparecerían en botones, los cuales cambiarían de color cada que se seleccionan

Requerimientos no funcionales:

1. Hacer la conexión de front-end y back-end mediante el uso de los endpoints asignados para este apartado.
2. La información se envía mediante la API REST para que pueda recibir el token de autenticación del usuario que está solicitándola
3. Los datos se guardan automáticamente en el back, por lo cual no es necesario tener un botón de guardar.

Desarrollo:

Una vez teniendo estos datos se procedió a realizar la conexión del back, se revisó los endpoints a utilizar, el primero seria para mostrar los temas seleccionados anteriormente

y el segundo sería para guardar los cambios al momento de seleccionar o deseleccionar los botones.



Ilustración 23 Endpoint de Social Interests en Hoppscotch

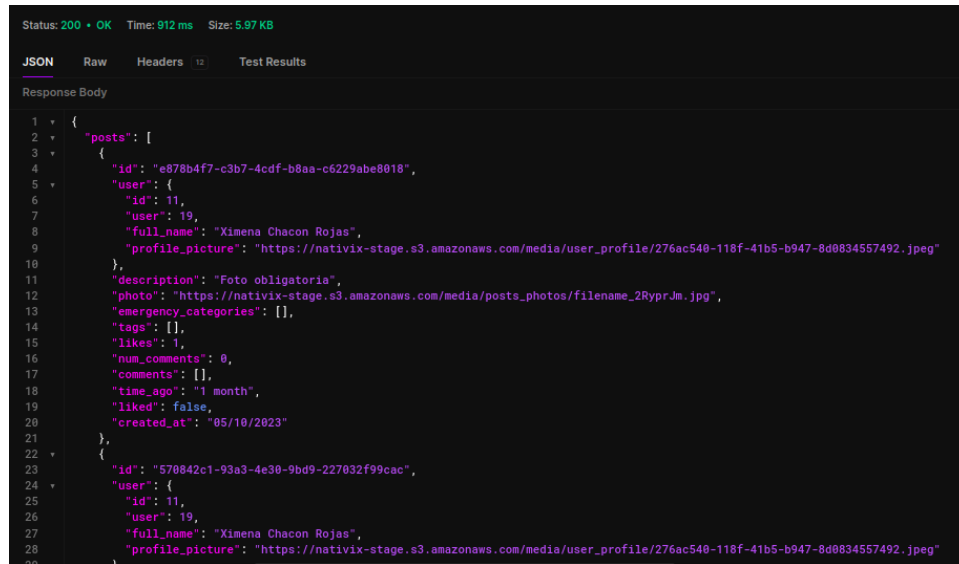


Ilustración 24 Interfaz del endpoint a utilizar

3.3.5.3 Agregar las conexiones del back en la pantalla del perfil de los usuarios de “Followers y Following”

Para esta tarea únicamente fue necesario realizar la conexión del back, anteriormente ya habíamos hecho algo parecido en el módulo de “Community”, cuando ingresábamos al perfil de alguien que buscábamos. Para esta pantalla se reutilizaron endpoints, lo cual hizo más rápida esta tarea.

1. Se integro el endpoint de la información del usuario seleccionado, el endpoint de las publicaciones que este había realizado y el endpoint de las propiedades que este tuviera.
2. Igualmente se agregó funcionalidad al botón de follow/unfollow con el endpoint de seguir a las personas para que estos cambios se reflejarán en la lista de seguidos

3.3.6 Sprint 6: Cambios solicitados por el cliente, mejora de usabilidad en la pantalla de “Crear publicaciones”

En este sprint se dedicó más a realizar pequeños que el cliente había solicitado y mejorar algunos aspectos de la pantalla de “crear publicaciones” para así mejorarla experiencia del usuario al usar esta pantalla.

3.3.6.1 Cambios solicitados por el cliente

Cuando se le mostraron los avances al cliente anteriormente se solicitó realizar un pequeño cambio de diseño en el apartado de publicación, el cual consistía en reacomodar los botones de like y comentarios junto con sus etiquetas, tanto en la sección de community como en la de mis publicaciones.

El cambio quedo de la siguiente manera



Ilustración 25 Cambios solicitados por el cliente en la sección de Posts (Antes y Después)

3.3.6.2 Mejora de usabilidad en la pantalla de “Crear publicaciones”

En esta pantalla se agregaron pequeños cambios para poder mejorar la experiencia del usuario al momento de usarla. Las modificaciones realizadas fueron:

1. Agregar nombre a los botones para saber que categoría contenía, ya que solo eran iconos de flechas y sabías el contenido de estos hasta el momento de abrirlos. De igual manera cuando seleccionaras alguna opción este botón cambiaría de nombre por la selección hecha.



Ilustración 26 Cambios solicitados por el cliente en la sección de crear un post

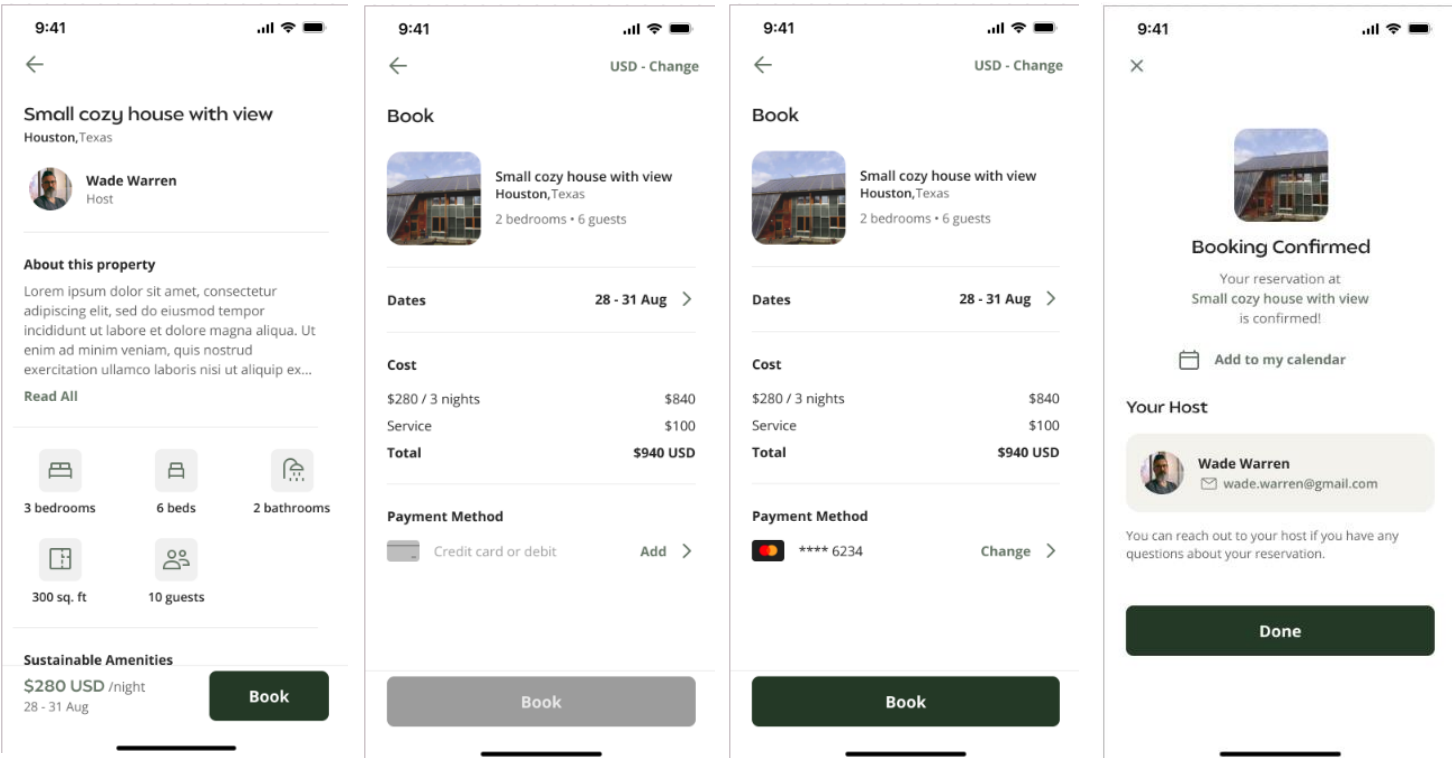
2. Deshabilitar el botón de publish. En esta pantalla la única cosa obligatoria al momento de crear la publicación era la foto, entonces era necesario hacer que el botón no funcionara si no había foto, se cambió de color para mostrarlo inactivo, cuando el usuario agregaba la foto este cambiaría de color

3.3.7 Sprint 7: Crear reservas de las propiedades desde los perfiles del módulo de “Community” y el módulo de “Followers y Following”, entrar al perfil de los usuarios seleccionando su nombre desde la lista de publicaciones

Para este sprint fue necesario agregar nuevas pantallas a módulos trabajados anteriormente, y una nueva función a la pantalla de publicaciones

3.3.7.1 Crear reservas de las propiedades desde los perfiles del módulo de “Community” y el módulo de “Followers y Following”

Para trabajar con esta nueva tarea en estos módulos fue necesario revisar el diseño de reservas en zeplin



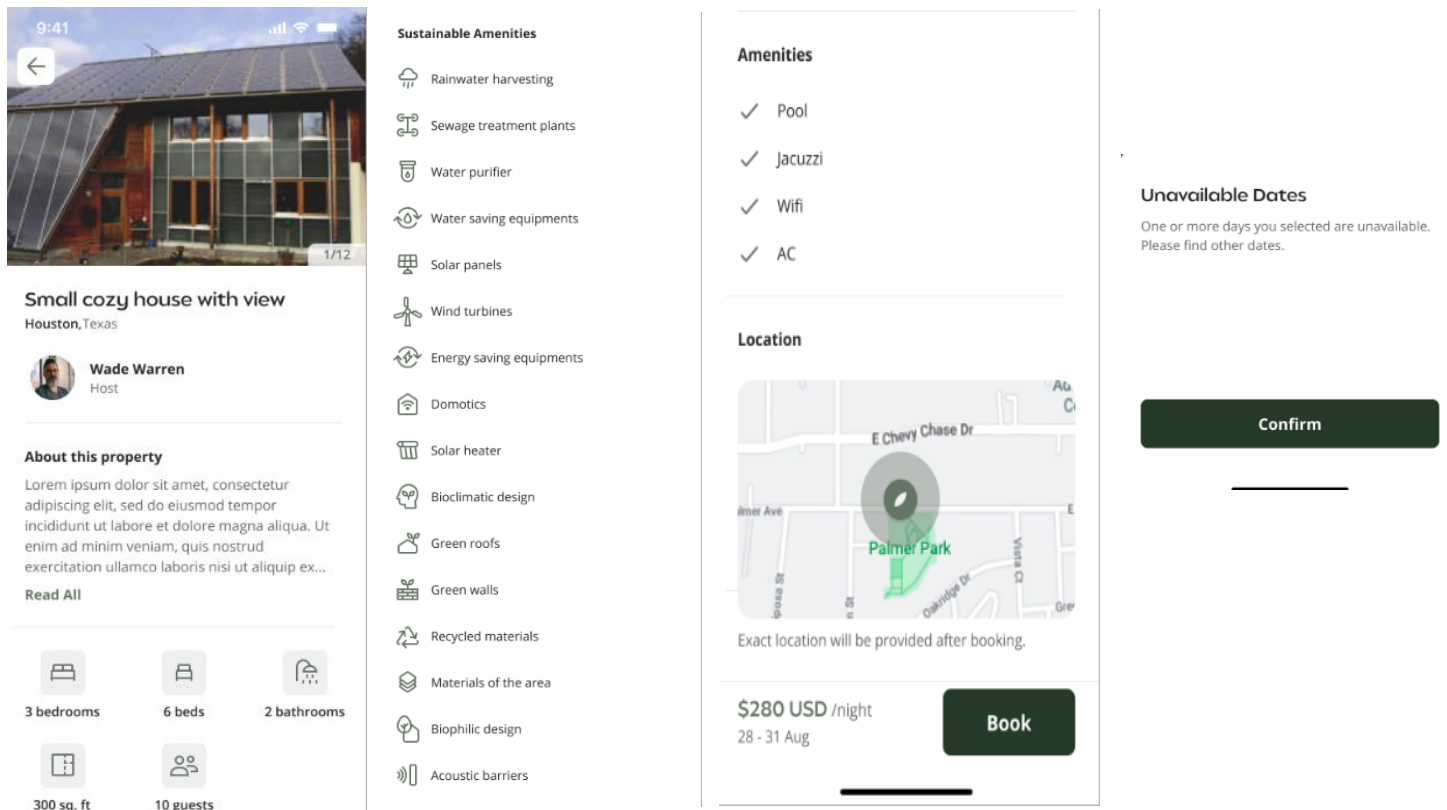


Ilustración 27 Maquetación de las pantallas de "Create Reservations" en zeplin

Anteriormente esta sección ya había sido implementada en otro modulo, asi que reutilizaríamos componentes y de esta manera tener un código más limpio y eficiente. Igual al reutilizar códigos automáticamente se harían las conexiones de front-back

Requerimientos funcionales:

1. El usuario podría entrar a ver los detalles de la propiedad del usuario que busco o de cualquier usuario que tuviera en sus listas de followers y following
2. Si al usuario le llamaba la atencion podria hacer una reserva desde aquí sin necesidad de buscarla en el módulo de "search properties"
3. Para hacer una reserva el usuario tendría un botón en la parte inferior de la descripción de la propiedad para que este lo llevara a las pantallas correspondientes.
4. En la descripción de la propiedad podrá ver las fotos, el nombre del host, las características de la casa y la ubicación en el mapa y el precio por noche
5. Ya en las pantallas de reserva podría seleccionar fecha, agregar algún método de pago en caso de no tener y poder reservar la propiedad.

Requerimientos no funcionales:

1. Hacer la conexión de front-end y back-end mediante el uso de los endpoints asignados para este apartado.

2. La información se envía mediante la API REST para que pueda recibir el token de autenticación del usuario que está solicitándola
3. Mostrar un mensaje en caso de que la fecha seleccionada este reservada por alguien mas
4. Toda la información debe de estar protegida ya que se manejan datos sensibles

Desarrollo:

En el código se ubicaron los componentes para poder agregarlos al módulo de community y followers y following.

Una vez localizados y agregados al módulo correspondiente no fue necesario hacer alguna conexión ya que la conexión ya estaba hecha en el módulo de “search properties”

CAPITULO IV: RESULTADOS

4.1 RESULTADOS

Una vez terminadas las secciones que estuvieron asignadas a mi cargo, estas quedaron de la siguiente manera en la aplicación:

4.1.1 “Followers y Following”, “My Publications”, “Payment methods”

4.1.1.1 “Followers y Following”

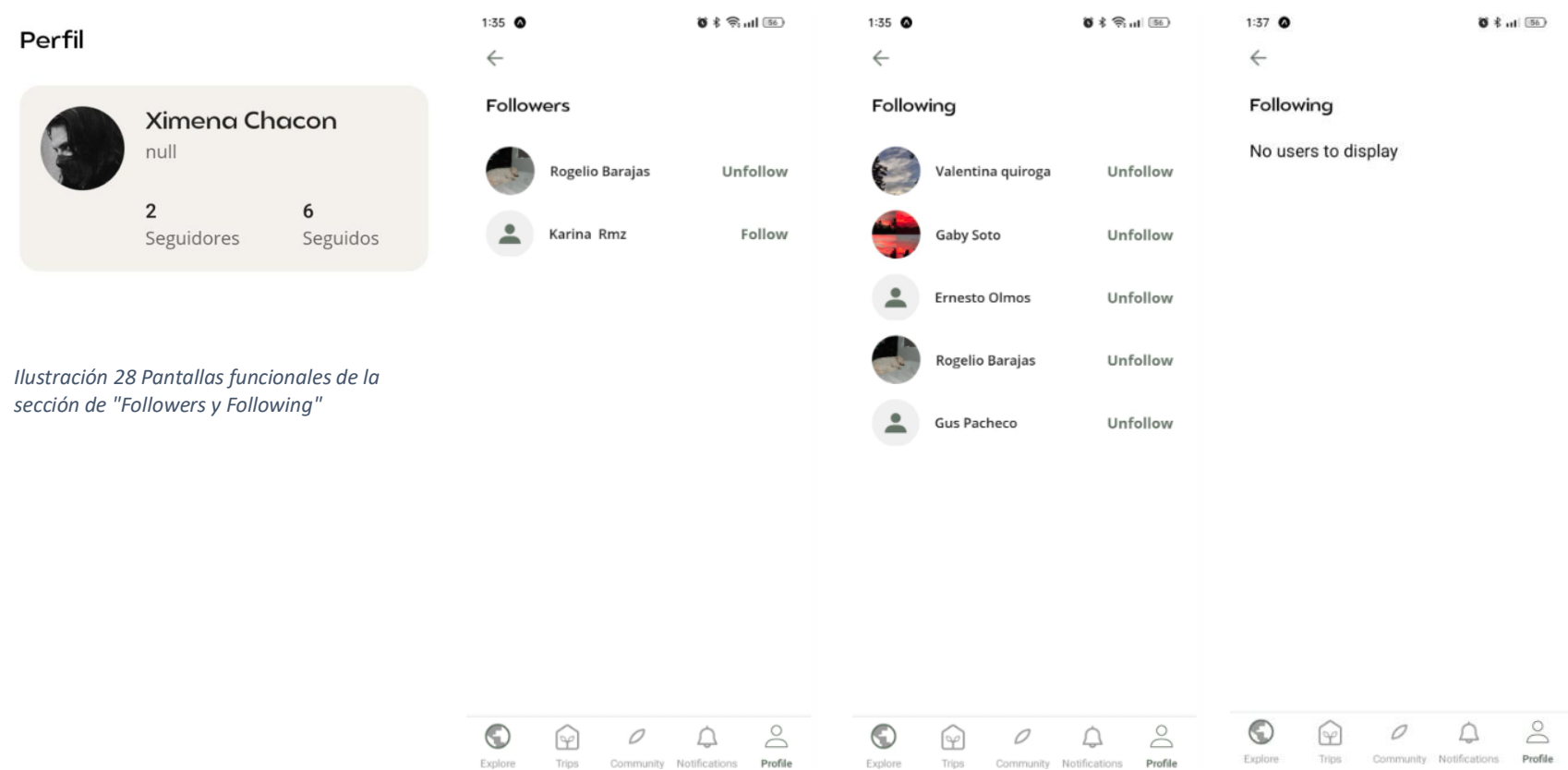


Ilustración 28 Pantallas funcionales de la sección de "Followers y Following"

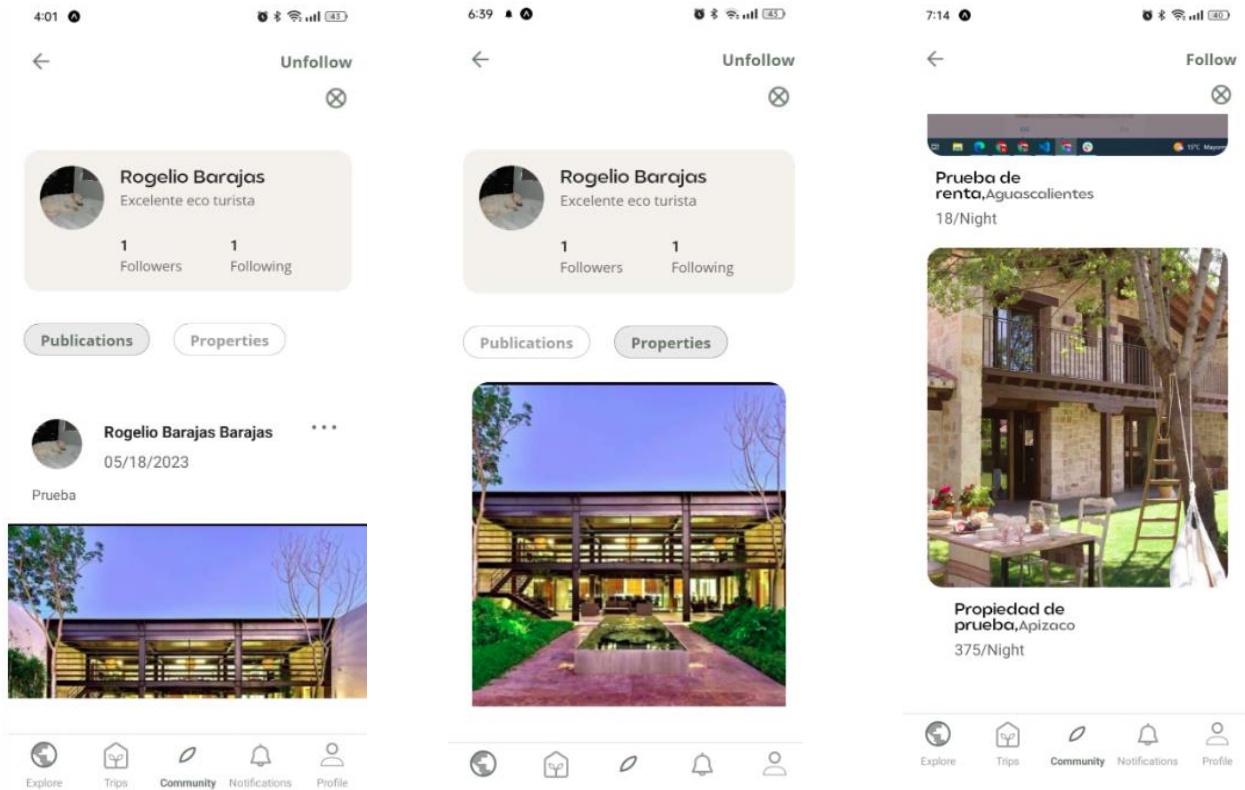


Ilustración 29 Pantallas funcionales de los perfiles desde "Followers y Following"

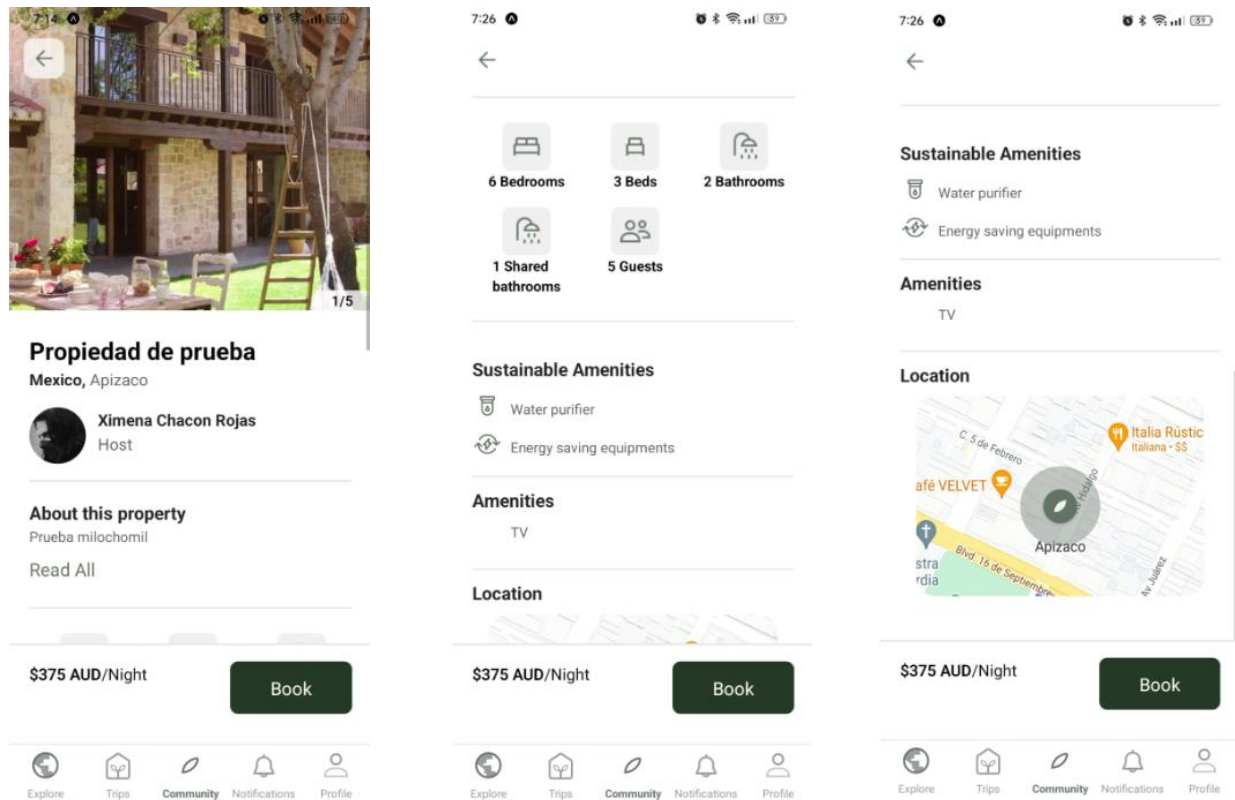


Ilustración 30 Pantallas funcionales de crear reservaciones desde los perfiles

4.1.1.2 “My Publications”

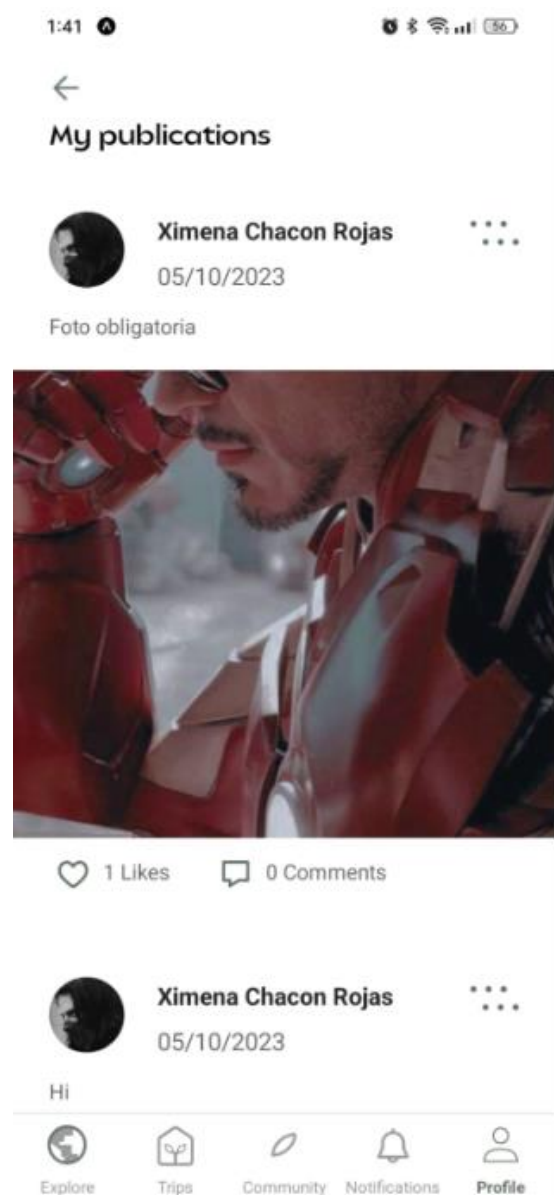


Ilustración 31 Pantallas funcionales de la sección "My publications"

4.1.1.3 “Payment methods”

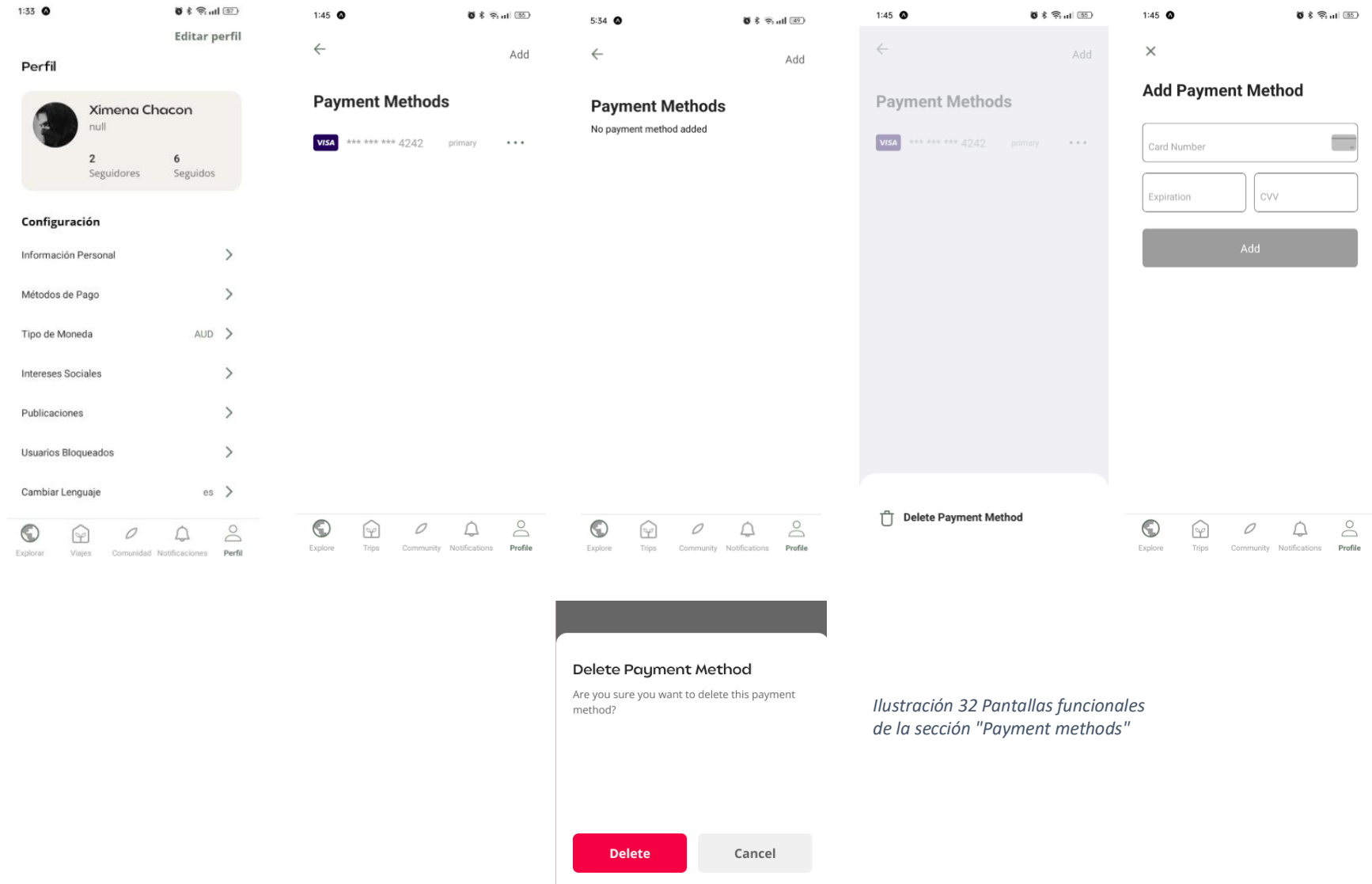


Ilustración 32 Pantallas funcionales de la sección "Payment methods"

4.1.2 Módulo de “Privacy Policy”, Módulo de “Community”

4.1.2.1 “Privacy Policy”

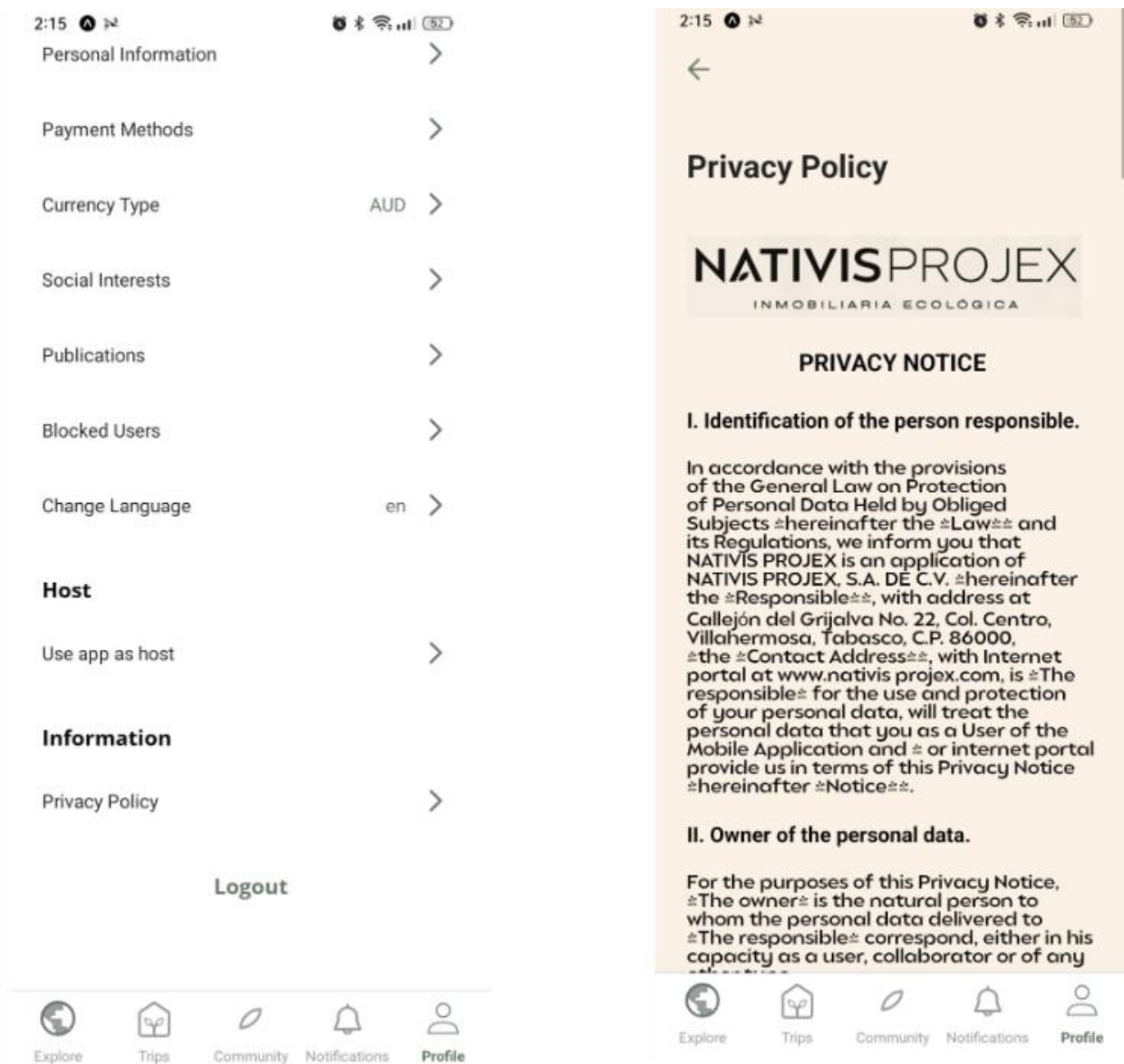


Ilustración 33 Pantallas funcionales de la sección "Privacy Policy"

4.1.2.2 “Community”



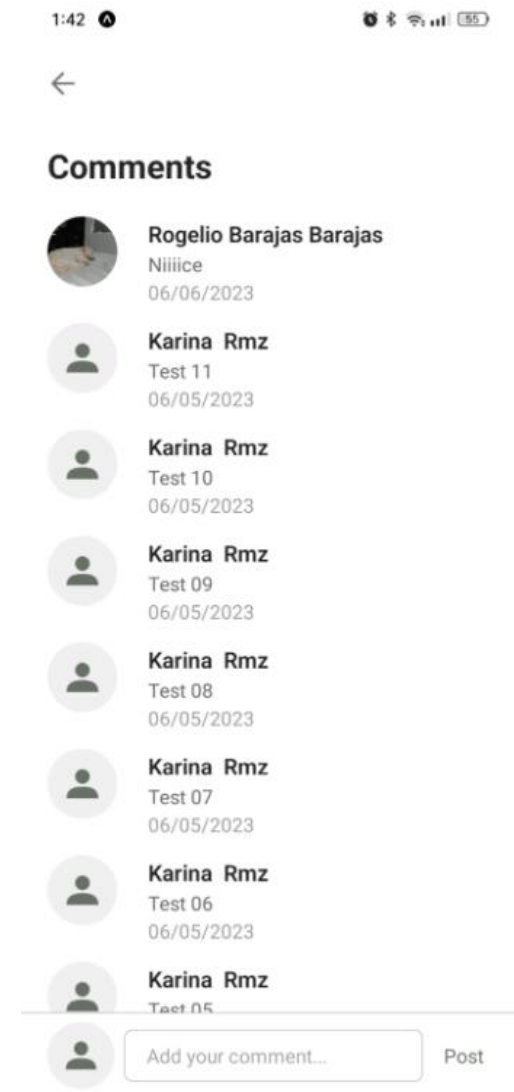
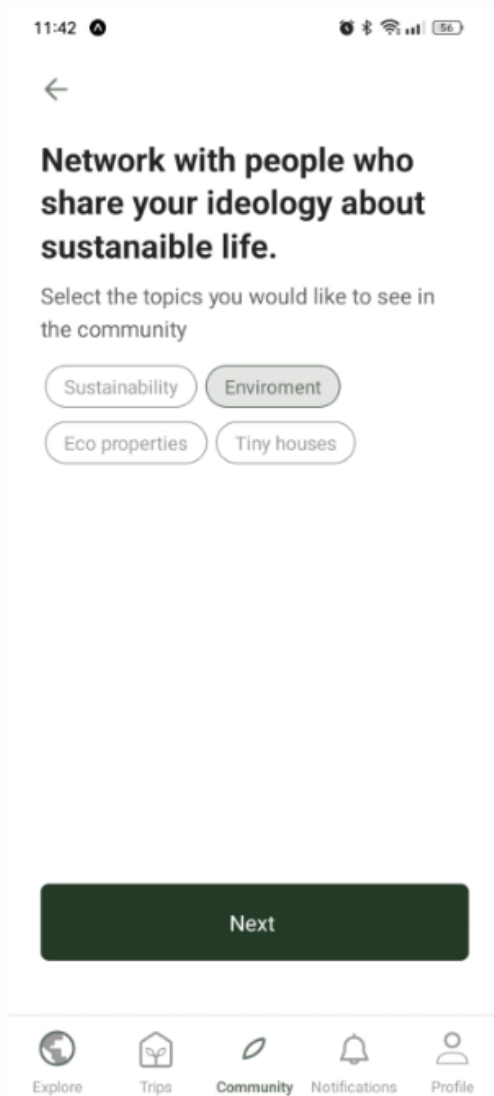


Ilustración 34 Pantallas funcionales de la sección "Community" apartado de posts

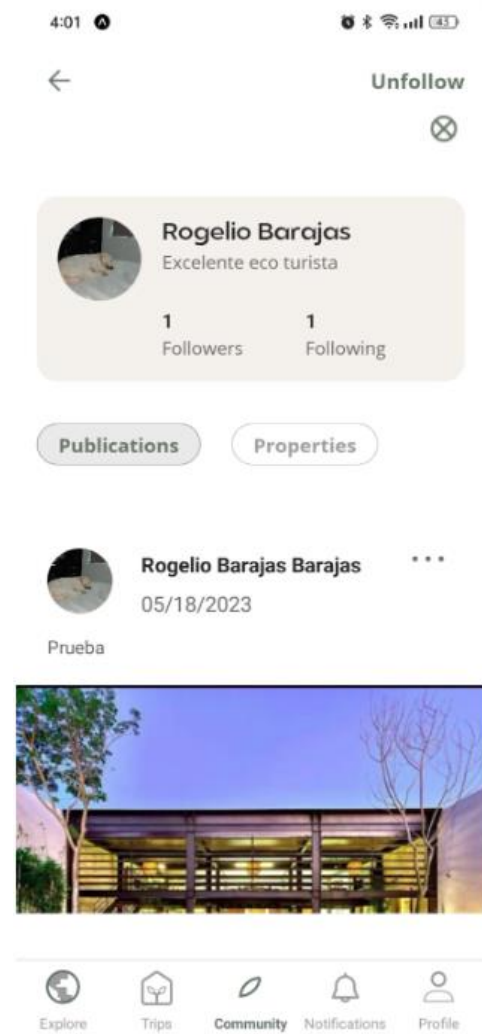
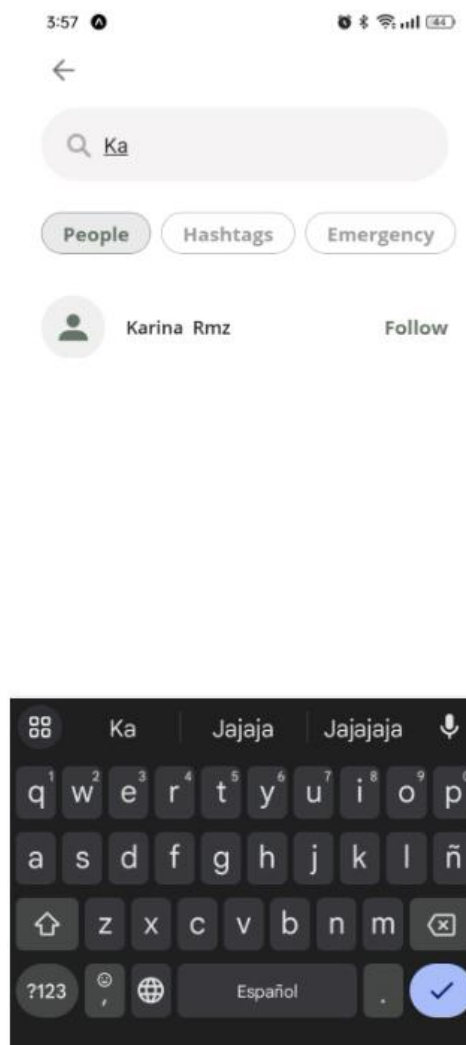
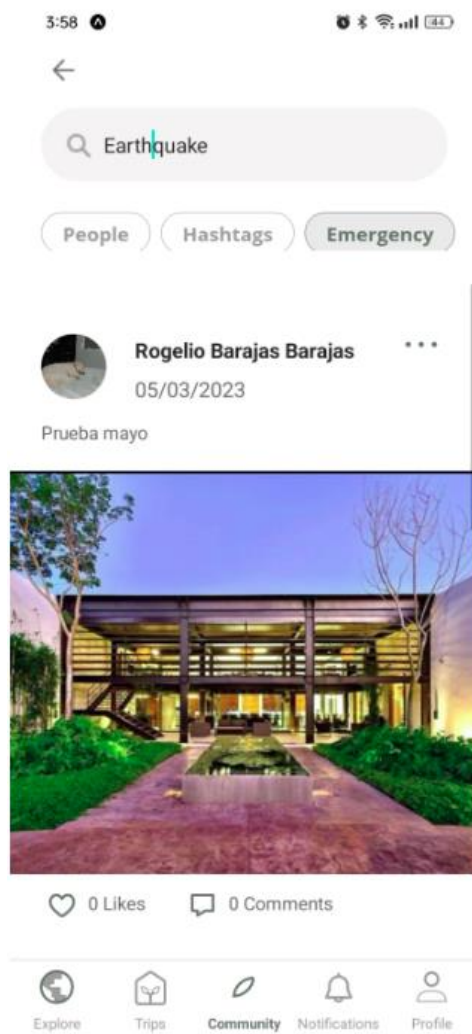


Ilustración 35 Pantallas funcionales de la sección "Community" apartado de búsqueda

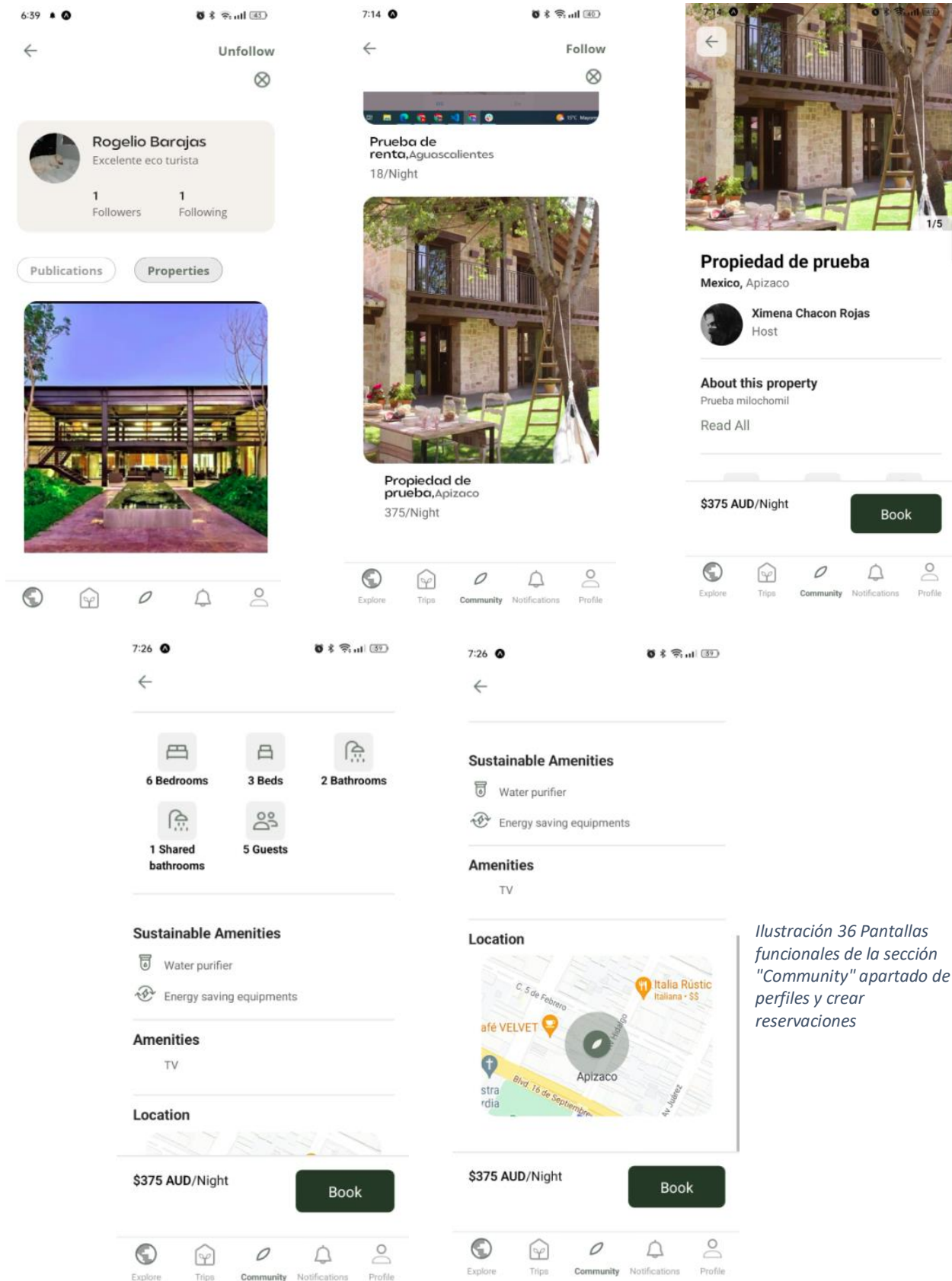


Ilustración 36 Pantallas funcionales de la sección "Community" apartado de perfiles y crear reservaciones

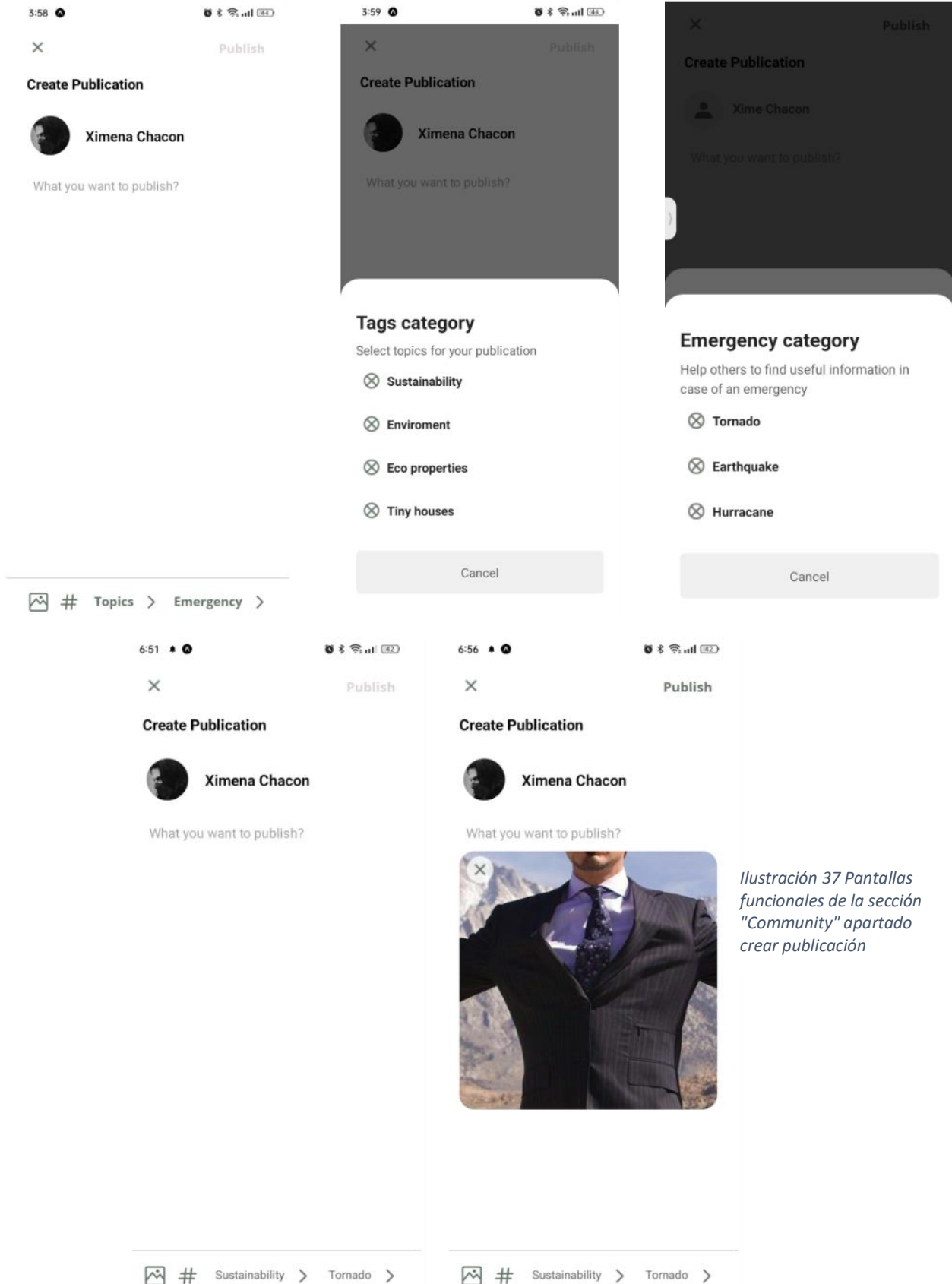
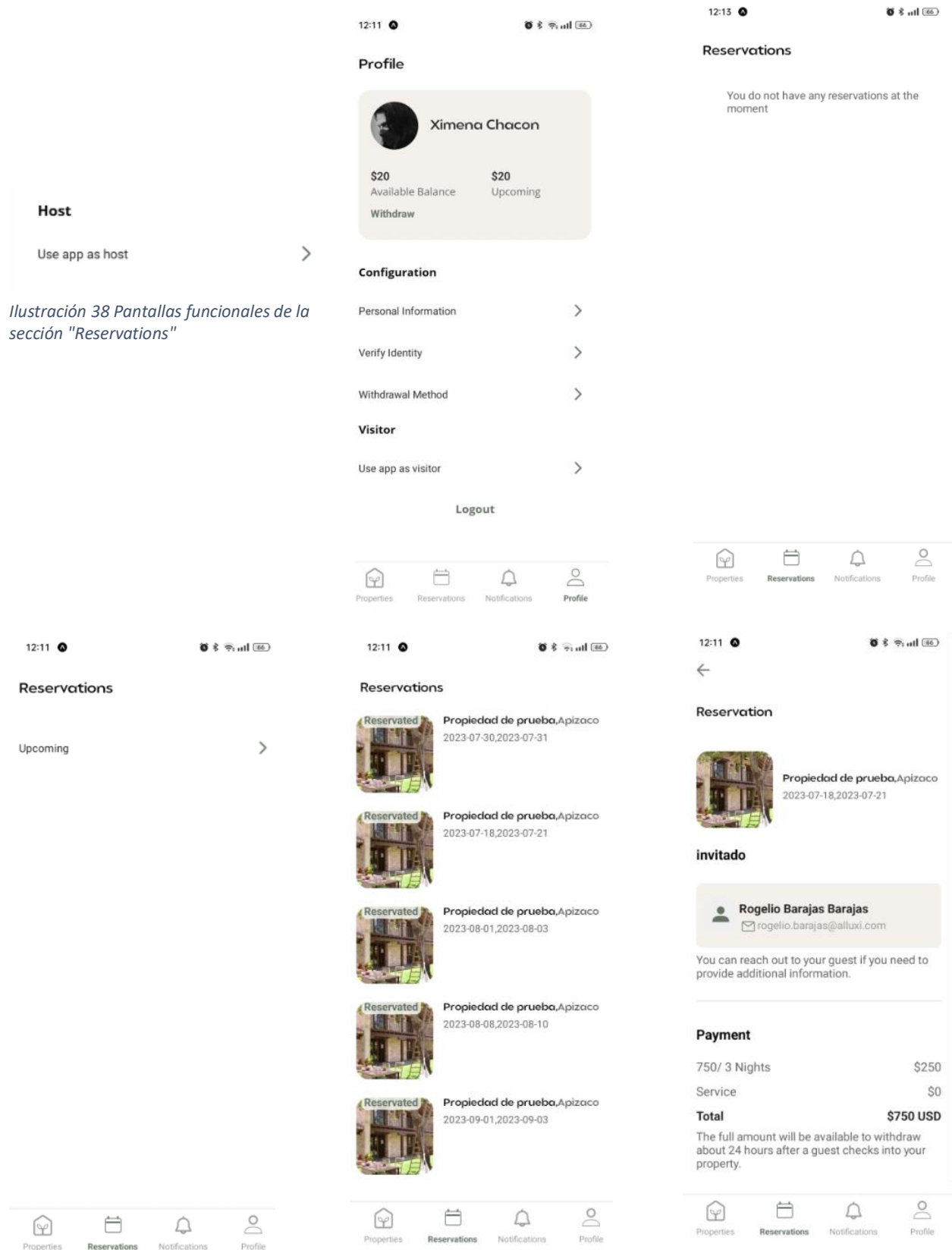


Ilustración 37 Pantallas funcionales de la sección "Community" apartado crear publicación

4.1.3 Módulo de “Reservations”

4.1.3.1 “Reservations”



4.1.3.2 “Social interests”

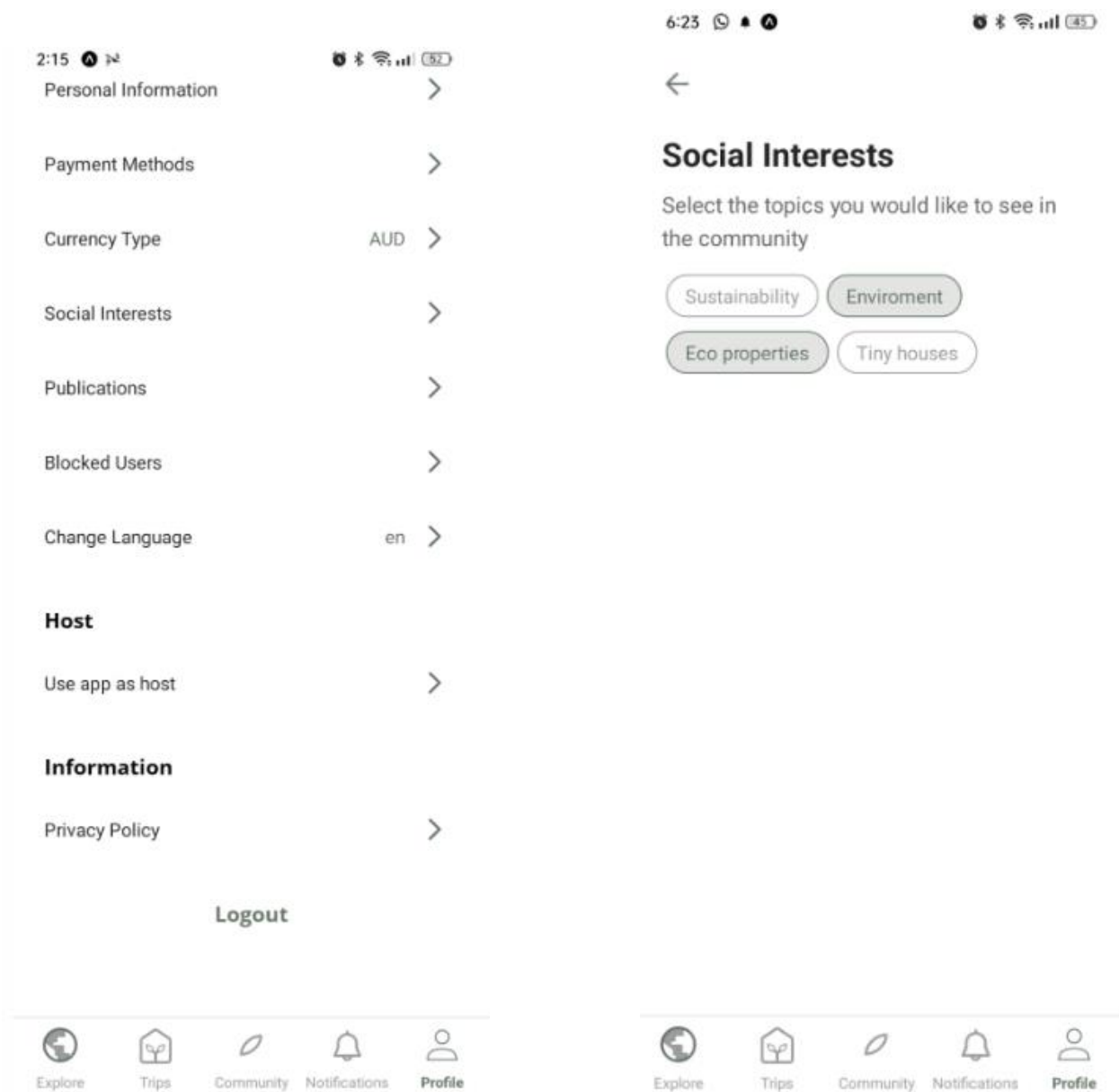


Ilustración 39 Pantallas funcionales de la sección "Social Interests"

CAPITULO V: CONCLUSIONES

5.1 CONCLUSIONES DE PROYECTO, RECOMENDACIONES Y EXPERIENCIA PERSONAL PROFESIONAL ADQUIRIDA

5.1.1 CONCLUSIONES DE PROYECTO

El uso de React Native para el desarrollo de una aplicación puede ser una excelente opción si se requiere un desarrollo rápido, multiplataforma y si el rendimiento no es un factor crítico. Sin embargo, es importante evaluar las necesidades específicas de la aplicación y considerar otras opciones (como el desarrollo nativo) si se requiere un rendimiento excepcional o características muy avanzadas.

La comunidad de React Native es muy activa y ofrece una amplia gama de bibliotecas y componentes predefinidos que pueden acelerar el desarrollo de la aplicación. Esto permite a los desarrolladores aprovechar soluciones existentes en lugar de construir todo desde cero.

En conclusión, el proyecto de desarrollo de una aplicación de venta y renta de casas ecológicas utilizando tecnología React Native ofrece una solución innovadora y sostenible para aquellos que buscan viviendas más respetuosas con el medio ambiente.

A través de la aplicación, los usuarios podrán acceder de manera fácil y rápida a una amplia variedad de casas ecológicas disponibles en diferentes ubicaciones. La tecnología React Native garantiza una experiencia de usuario fluida y una interfaz intuitiva, lo que facilita la navegación y la búsqueda de propiedades. Además, la aplicación fomenta la adopción de prácticas amigables con el medio ambiente al promover la venta y renta de casas con características sostenibles, como paneles solares, sistemas de recolección de agua de lluvia y materiales de construcción eco-amigables. Esto permite a los usuarios encontrar viviendas que se ajusten a sus valores y contribuir a reducir el impacto negativo en el entorno.

El proyecto de desarrollo de esta aplicación brinda una solución integral para el mercado inmobiliario ecológico, mejorando la accesibilidad y la conciencia ambiental de los usuarios. Promueve un estilo de vida más sustentable y consciente, al mismo tiempo que impulsa la adopción de tecnologías modernas como React Native en la industria.

5.1.2 EXPERIENCIA PERSONAL / PROFESIONAL ADQUIRIDA

Durante los meses en los que realicé mis residencias profesionales en ALLUXI, tuve una experiencia personal muy agradable. El ambiente de trabajo con mis compañeros fue excelente, siempre obtuve ayuda de todo el equipo, desde el jefe de la empresa hasta los Project Manager y los miembros del equipo de desarrollo de la aplicación. Esto hizo que el trabajo no se sintiera abrumador, ya que siempre había alguien dispuesto a darme una mano. Gracias a esto, puedo afirmar que no fue difícil integrarme a los equipos de trabajo.

Hablando en términos profesionales, adquirí conocimientos en una nueva tecnología que no había utilizado antes. Además, mejoré mis habilidades de comunicación con los clientes, aprendiendo a escuchar y entender sus necesidades. Estas habilidades son fundamentales si queremos dedicarnos a este campo. Al ingresar a ALLUXI, pude reforzar los conocimientos que ya tenía mediante las tareas que realicé. Esto fue muy beneficioso, ya que pude poner en práctica lo que me habían enseñado en la institución.

CAPITULO VI: COMPETENCIAS DESARROLLADAS

6.1 COMPETENCIAS DESARROLLADAS Y/O APLICADAS.

Las competencias que yo adquirí y que apliqué durante este tiempo de residencias fueron las siguientes:

- **Capacidad de trabajo en equipo:** Es fundamental tener la habilidad de interactuar de manera asertiva con los miembros del equipo. Durante el desarrollo de un proyecto, trabajarás de forma continua con ellos, evitando así malos entendidos.
- **Iniciativa/Proactividad:** Esta competencia es importante, ya que, aunque se te asignen tareas con características específicas, siempre se agradece tener la iniciativa de solucionar errores en la aplicación, incluso si no te lo indicaron. De igual manera, si encuentras errores en el trabajo de otros miembros del equipo, siempre es bien recibido informarles o incluso resolverlos sin necesidad de esperar instrucciones.
- **Capacidad de aprendizaje:** Es importante saber aprender de los errores que cometamos y adquirir nuevos conocimientos para mejorar en nuestro trabajo, ya sea en programación, diseño, gestión de proyectos, etc. Esta habilidad siempre será valorada por las empresas.
- **Flexibilidad:** Al igual que las demás competencias, esta también es fundamental. Es importante tener la mente abierta y estar dispuesto a los cambios. Durante reuniones con clientes, es común que surjan modificaciones, incluso si ya se habían acordado requisitos previamente. Además, no conformarnos con lo que hacemos y salir de nuestra zona de confort siempre que sea necesario es esencial.
- **Comunicación:** Esta competencia va de la mano con todas las demás, ya que la clave para que las cosas funcionen correctamente es mantener una buena comunicación, ya sea con clientes, compañeros de trabajo o superiores. La comunicación es clave para el éxito.
- **Responsabilidad:** Esta competencia es importante en el ámbito laboral. Incluye desde ser responsables con los horarios de trabajo hasta cumplir con las entregas de forma puntual. Ser una persona responsable habla bien de nosotros y de nuestro trabajo.

CAPITULO VII: FUENTES DE INFORMACIÓN

7.1 FUENTES DE INFORMACIÓN

Alex. (2019, mayo 31). *GitKraken*. Medium.

<https://alexmarket.medium.com/gitkraken-ea27eb8e8301>

Ancheta, W. (s/f). *Desarrollo Más Fácil React Native con Expo*. Code Envato Tuts+.

Recuperado el 21 de agosto de 2023, de <https://code.tutsplus.com/es/easier-react-native-development-with-expo--cms-30546t>

Bello, J. L. (2021, octubre 23). *Android: ¿Qué es un emulador y cuáles son las mejores opciones?* FayerWayer.

<https://www.fayerwayer.com/moviles/2021/10/23/android-que-es-un-emulador-y-cuales-son-las-mejores-opciones/>

Carrillo, P. (2022, julio 11). *React Native: ¿Qué es?* Doonamis.

<https://www.doonamis.es/react-native-que-es-ventajas-desventajas/>

¿Cómo desarrollar apps multiplataforma? (2021, diciembre 7). *KeepCoding Bootcamps*.

<https://keepcoding.io/blog/desarrollar-aplicaciones-moviles-multiplataform/>

Eduardo. (2022, agosto 21). *¿Qué es una aplicación móvil?* Anincubator; Anincubator

LLC. <https://anincubator.com/que-es-una-aplicacion-movil/>

Fernández, C. (2021, marzo 17). *Apps multiplataforma. Qué son y características.*

ABAMobile. <https://abamobile.com/web/apps-multiplataforma-que-son-y-caracteristicas/>

Flores, F. (2022, julio 22). Qué es Visual Studio Code y qué ventajas

ofrece. *Openwebinars.net*. <https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/>

Gustavo, B. (2019, abril 12). *Qué es GitHub y cómo empezar a usarlo*. Tutoriales

Hostinger. <https://www.hostinger.mx/tutoriales/que-es-github>

Ontiveros, F. (2017, abril 26). *¿Por qué desarrollar apps de React Native en*

Expo? Leopark Lab. <https://medium.com/leopark-lab/por-qu%C3%A9-desarrollar-apps-de-react-native-en-expo-2e1b83e4d00a>

Pérez, C. (2022, abril 26). *Hoppscotch: Herramienta para API Testing de Código*

Abierto. Blog de Testing y Calidad de Software | Abstracta Chile; Abstracta Chile. <https://cl.abstracta.us/blog/hoppscotch-herramienta-api-testing/>

¿Qué es API? Ejemplos, ventajas y tipos. (2022, febrero 22). Blog SYDLE.

<https://www.sydle.com/es/blog/api-6214f68876950e47761c40e7>

¿Qué es React Native? (2019, marzo 13). Deloitte Spain.

<https://www2.deloitte.com/es/es/pages/technology/articles/que-es-react-native.html>

(S/f-a). Amazon.com. Recuperado el 21 de agosto de 2023, de

<https://aws.amazon.com/es/what-is/javascript/>

(S/f-b). Medium.com. Recuperado el 21 de agosto de 2023, de

<https://mauritorra.medium.com/zeplin-el-%C3%A1gora-para-dise%C3%B1adores-y-desarrolladores-11db2fa8d36>

(S/f-c). Iodocs.com. Recuperado el 21 de agosto de 2023, de

<https://www.iodocs.com/es/que-son-las-api/>

(S/f-d). Amazon.com. Recuperado el 21 de agosto de 2023, de

<https://aws.amazon.com/es/what-is/api/#:~:text=API%20de%20REST.-,API%20de%20REST,datos%20de%20salida%20al%20cliente.>

ANEXOS

CARTA DE NO INCONVENIENCIA



Monterrey, N.L. a 28 de Junio de 2023

Asunto: Carta de No inconveniencia.

DRA. YESICA IMELDA SAAVEDRA BENITEZ
DIRECTORA DEL INSTITUTO TECNOLÓGICO DE
APIZACO
Presente. -

Por medio de la presente me dirijo a usted, para comunicarle que la C. **XIMENACHACON ROJAS**, con número de control 18370930, pasante de la carrera de **Ingeniería en Tecnologías de la Información y Comunicaciones** del Instituto Tecnológico de Apizaco, terminó satisfactoriamente su proyecto de Residencias Profesionales denominado: **DESARROLLO APP/MÓVIL Proyecto NativisProjex** en el período comprendido del **23 de Enero del 2023 al 23 de Mayo del 2023**.

Tiene autorizado presentarlo para su **TITULACIÓN INTEGRAL INFORME DE RESIDENCIAS PROFESIONAL**, confiando que la presentación de la información del Proyecto se realizará de manera responsable.

A petición del interesado y para fines legales que al interesado convengan, se extiende la presente carta de No inconveniencia al día veinticuatro del mes de Mayo del dos mil veintitrés.

Atentamente,

Lic. Andrea Verónica Mata de Orozco
Departamento Administrativo y R. Humanos Teléfono
Celular 818 1622336

Alluxi, S.A. de C.V.

CARTA DE LIBERACIÓN DE LA EMPRESA



Monterrey, N.L. a 22 de Junio del 2023

Asunto: **Carta de Liberación Residencias Profesionales**

DRA. YESICA I. SAAVEDRA BENITEZ

Directora del Campus

INSTITUTO TECNOLÓGICO DE APIZACO

Presente

Por medio de la presente, me dirijo a usted para hacer de su conocimiento que

La C. CHACÓN ROJAS XIMENA alumna de INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN con número de matrícula 18370930 ha concluido satisfactoriamente su Residencia Profesional en esta Empresa que represento, desarrollando el proyecto denominado:

“DESARROLLO APP/MOVIL Proyecto NativisProjex”, en el período comprendido del 23 de Enero del 2023 al 23 de Mayo del 2023 cubriendo el total de 500 horas respectivas.

Sin otro particular, se extiende la presente para los fines que al interesado convenga.

Atentamente,

A. Verónica Mata de Orozco

Depto. Administrativo y R.H.

Tel. Cel. 818 1622336