

Universidad ORT Uruguay

Segunda Entrega

Interacción Humano - Computadora

Docentes:

Luis Barragué

Bruno Ferrari

Noelia Bentancor 242970

Sofía Decuadra 233397

Agustín Ferrari 240503

2022

Índice

Declaración de autoría	5
Introducción	6
Repositorios	6
Tecnologías y herramientas de desarrollo	6
Alcance	7
Producto	7
Propuesta de valor	7
Análisis del problema	7
Especificación de requerimientos	7
Cambios realizados a la propuesta original	10
Proceso	10
Gestión de proyecto	10
Release plan	10
Sprint 0	10
Sprint 1	11
Sprint 2	11
Sprint 3	12
Gestión de riesgos	12
Gestión de la configuración	13
Control de cambios y de versiones	13
Descripción de la arquitectura	14
Vista de módulos	14
Decisiones de diseño	15
Diseño de cada componente, mecanismos de comunicación, estructura de datos	15
Expensify invitations	16
S3	16
	2

App	16
Backend	16
MySQL Database:	16
Expo Notifications API y Firebase	16
Modelo de tablas	17
Justificaciones de la solución	18
Autenticación y autorización	18
Seguridad	19
Portabilidad	19
Modificabilidad y desacoplamiento	19
Diseño de UI/UX	21
Coincidencia entre el sistema y el mundo real	21
El usuario tiene control y libertad	21
Consistencia y estándares	21
Prevención de errores	21
Reconocer en lugar de recordar	22
Estética y diseño minimalista	22
Ayudar al usuario a reconocer, diagnosticar y recuperarse de los errores	22
Justificaciones de Clean Code	22
Mantener consistencia	22
Comentarios	23
Estructura del código fuente	23
Pruebas	23
Cobertura de pruebas	24
Retrospectiva general	25
Instructivo de instalación	26
Backend	26

App	26
Aplicación de principios arquitectónicos específicos de React Native	27
Componentes	27
Navegación	27
Hooks	28
Context	28
Queries	28
Anexo	29
Anexo 1: Historias de usuario	29
Anexo 2: Prototipo	39
Anexo 3: Aplicación	53
Anexo 4: Especificación de la API	74
Anexo 5: Archivo .env	76

Declaración de autoría

Nosotros, Noelia Bentancor, Sofía Decuadra y Agustín Ferrari, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos la materia Interacción Humano - Computadora;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;

Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Introducción

El presente documento tiene como objetivo informar acerca del alcance, el proceso y las decisiones tomadas para el desarrollo de una aplicación mobile para el manejo de gastos de una familia.

Repositorios

- Backend: [Link](#)
- Frontend: [Link](#)
- Web de invitaciones: [Link](#)

Tecnologías y herramientas de desarrollo

- **Backend:** Node js
- **Frontend:** Desarrollo *mobile* con React Native.

Alcance

Producto

El producto consiste en una aplicación para dispositivos Android e iOS para el manejo de gastos en una familia. Los usuarios podrán registrar sus compras (que serán compartidas con toda la familia) y consultar por gráficos donde se puede ver la evolución de los gastos a lo largo del tiempo.

Propuesta de valor

Mantener el registro de todos los gastos que se hacen en una familia permite identificar y eliminar hábitos de gastos que no son necesarios en su vida financiera. Este seguimiento les permitirá saber cuándo dejar de gastar en una categoría determinada (como ropa o comida por ejemplo) y encontrar servicios que realmente no están utilizando (como una membresía al gimnasio al cual ya no se está concurriendo).

Al ser más conscientes de sus hábitos, les ayudará a ahorrar y frenar gastos que arruinan su presupuesto, promoviendo así el alcance de sus objetivos financieros.

Análisis del problema

Especificación de requerimientos

Link a [Trello](#) con el tablero del equipo.

1. RF1: Registro de usuario administrador

Cualquier usuario puede registrarse en la aplicación como administrador de su familia. Para ello debe ingresar nombre, email, nombre de la familia (único en el sistema) y contraseña. Al registrarse habrá creado una familia.

2. RF2: Registro de usuario mediante invitación

Un usuario administrador puede invitar a otros usuarios administradores y/o miembros a la familia, enviándoles por algún medio (correo electrónico, mensaje por WhatsApp) con un link a la página de registro anterior sin el campo “nombre de familia”.

3. RF3: Autenticación de usuario

El usuario puede acceder a un formulario de ingreso que solicita su correo electrónico y una contraseña de acceso. Una vez autenticado el usuario, el sistema lo redireccionará a la pantalla de inicio para la familia de la que forma parte.

4. RF4: CRUD de categorías

RF4.1: Alta de categorías

El sistema permite a los usuarios administradores dar de alta categorías de gastos con los siguientes datos:

- Nombre (único por familia)
- Descripción (texto libre)
- Imagen que represente a la categoría (se puede sacar con la cámara del celular o cargar desde la galería)
- Presupuesto mensual de gastos (opcional)

RF4.2: Modificación de categorías

El sistema permite a los usuarios administradores modificar cualquier campo de la categoría (manteniendo la unicidad del nombre).

RF4.3: Baja de categorías

El sistema permite a los usuarios eliminar de forma lógica (para no perder gastos históricos) categorías.

5. RF5: Ver alerta por superar presupuesto mensual

Los usuarios administradores recibirán una notificación en su celular en caso de que se supere el presupuesto mensual de gastos para una categoría.

6. RF6: CRUD de gastos

RF4.1: Alta de gastos

El sistema permite a los usuarios administradores y miembros dar de alta gastos con los siguientes datos:

- Descripción (texto libre)
- Monto
- Fecha de producido
- Imagen de factura (se puede sacar con la cámara del celular o cargar desde la galería)

Además se registra de forma automática el usuario que creó el gasto y la fecha de registro.

RF4.2: Modificación de categorías

El sistema permite a los usuarios administradores modificar cualquier campo del gasto (menos el usuario que lo registró).

RF4.3: Baja de categorías

El sistema permite a los usuarios eliminar gastos.

7. RF7: Página de inicio familiar

Al ingresar a la aplicación los usuarios pueden ver una pantalla muestra los gastos registrados en lo que va del mes actual en una lista que indica el usuario que lo registró, la categoría, la fecha de producido y el monto. Si se selecciona un gasto, se debe mostrar el detalle de este: fecha de realizado, descripción e imagen de la factura.

8. RF8: Análisis de datos

Los usuarios administradores o miembros pueden ver en gráficos los gastos para un período dado agrupados por categoría.

Cambios realizados a la propuesta original

No se realizaron cambios a la propuesta original ya que consideramos que todas las funcionalidades descritas eran de suma importancia. Se realizó el **100%** de la funcionalidad acordada.

Proceso

Gestión de proyecto

Se utilizó una metodología ágil, específicamente SCRUM. Los requerimientos fueron entonces expresados como elementos en el Product Backlog y se fueron abordando en Sprints. Estos últimos teniendo una duración de 2 semanas, lo que dió un total de 3 Sprints hasta la fecha de entrega.

Ceremonias:

- *Periodic Meeting*: reunión al menos una vez a la semana para comentar avances y dificultades enfrentadas. Estas son una adaptación a las *Daily Meetings* definidas en SCRUM.
- *Sprint Review*: reunión al final de cada sprint para inspeccionar el incremento y adaptar el Product Backlog en caso de ser necesario.
- *Sprint Retrospective*: reunión al final de cada sprint para perfeccionar y ajustar el proceso de trabajo del equipo que realizamos.

Release plan

Sprint 0

Tareas	To Do	In Progress	Done	Estimación (SP)
Especificación de requerimientos			X	8
Definir tecnologías			X	3
Definir la arquitectura del backend			X	5
Definir template del			X	2

proyecto				
Gestión de proyecto			X	2
Realizar prototipo			X	3

Sprint 1

User Story	To Do	In Progress	Done	Estimación (SP)
#US1: Registro de usuario administrador			X	2
#US2: Enviar invitación			X	13
#US3: Aceptar invitación			X	5
#US4: Iniciar sesión			X	2
#US5: Cerrar sesión			X	2
#US7: Crear categoría			X	13
Total				37

Sprint 2

User Story	To Do	In Progress	Done	Estimación (SP)
#US6: Ver categorías			X	5
#US11: Agregar gasto			X	13
#US14: Página de inicio familiar			X	5
#US10: Ver alerta por superar presupuesto			X	13
Total				37

Sprint 3

User Story	To Do	In Progress	Done	Estimación (SP)
#US8: Modificar categoría			X	8
#US9: Eliminar categoría			X	3
#US12: Modificar gasto			X	8
#US13: Eliminar gasto			X	3
#US15: Filtros por período en gráfico			X	8
#US16: Análisis de gastos			X	8
Total				37

Gestión de riesgos

Los riesgos planteados al principio de este proyecto fueron los siguientes:

Riesgo	Plan de mitigación	Sprint/s que aplican
Historias de usuarios mal estimadas. No conocemos la velocidad del equipo.	Ajustar las historias de usuarios en cada sprint y volver a estimarlas en caso de que sea necesarios.	1, 2, 3
Dificultad en el aprendizaje de nuevas tecnologías.	Realizar cursos sobre React Native. Spikes de investigación.	1,2
Otros proyectos en paralelo.	Respetar las ceremonias y establecerlos horarios para poder trabajar a la par y avanzar.	3
Poca adherencia a SCRUM	Reflexionar sobre el proceso en las retrospectivas.	1, 2, 3

Nos enfrentamos efectivamente a situaciones que podrían haber puesto en riesgo el éxito del proyecto, como por ejemplo, la mala estimación de algunas historias. En el segundo sprint

particularmente, nos quedó una historia sin completar y con el equipo decidimos priorizar esta en el tercer sprint y realizarla en conjunto para luego seguir con el plan original, lo cual nos dio un buen resultado.

En adición, siendo que el emulador de Android es muy pesado, algunos de los integrantes debimos conseguir un celular para hacer las pruebas de manera más sencilla, ya que se precisa mucho espacio para poder correr la aplicación.

Gestión de la configuración

Control de cambios y de versiones

Se utilizaron dos repositorio en GitHub para el control de versiones siguiendo el modelo **Gitflow** para la creación de ramas, las cuales son nombradas según el tipo de cambio que agregan al repositorio:

- feature/[x]: rama que implementa la funcionalidad x
- fix/[x]: rama que arregla x
- refactor/[x]: rama que hace un refactor de x
- docs/[x]: rama que agrega documentación, siendo x el nombre del documento modificado/agregado

Cuando una rama es terminada se realiza un Pull Request a la rama develop.

Descripción de la arquitectura

Vista de módulos

Diagrama 1: Vista de descomposición

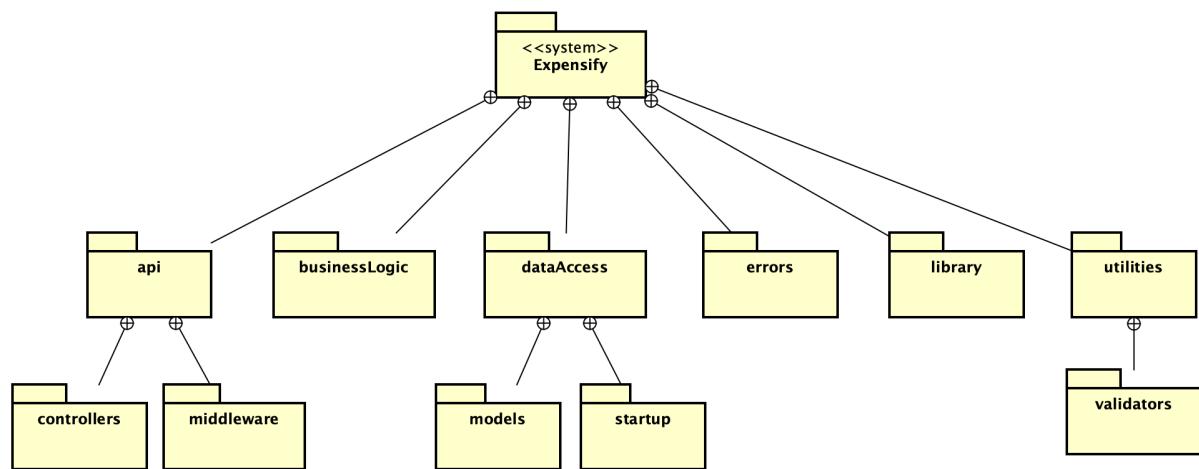
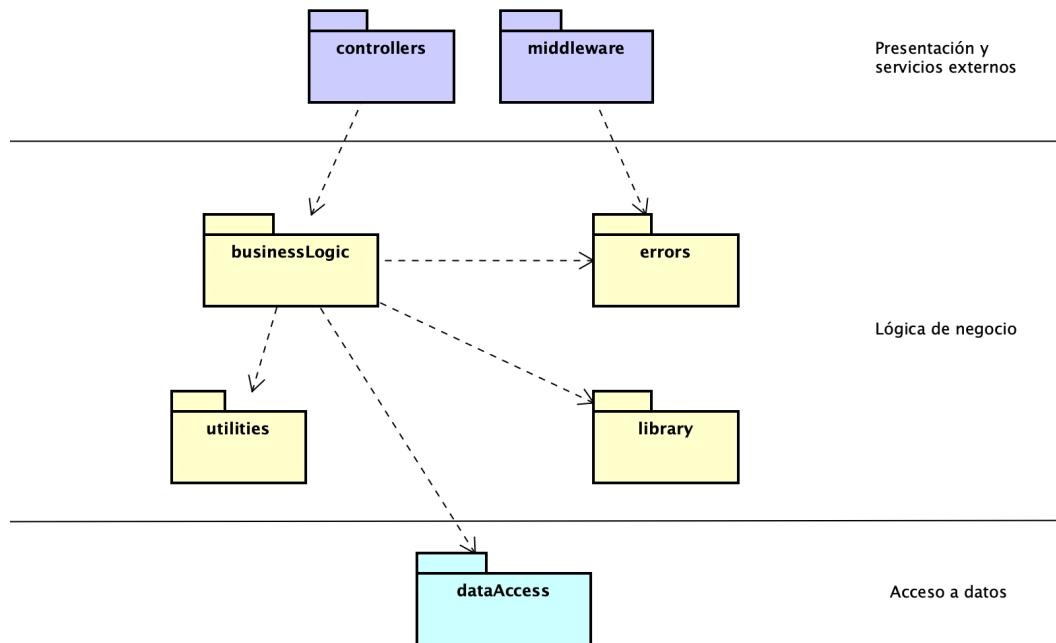


Diagrama 2: Vista de layers



Decisiones de diseño

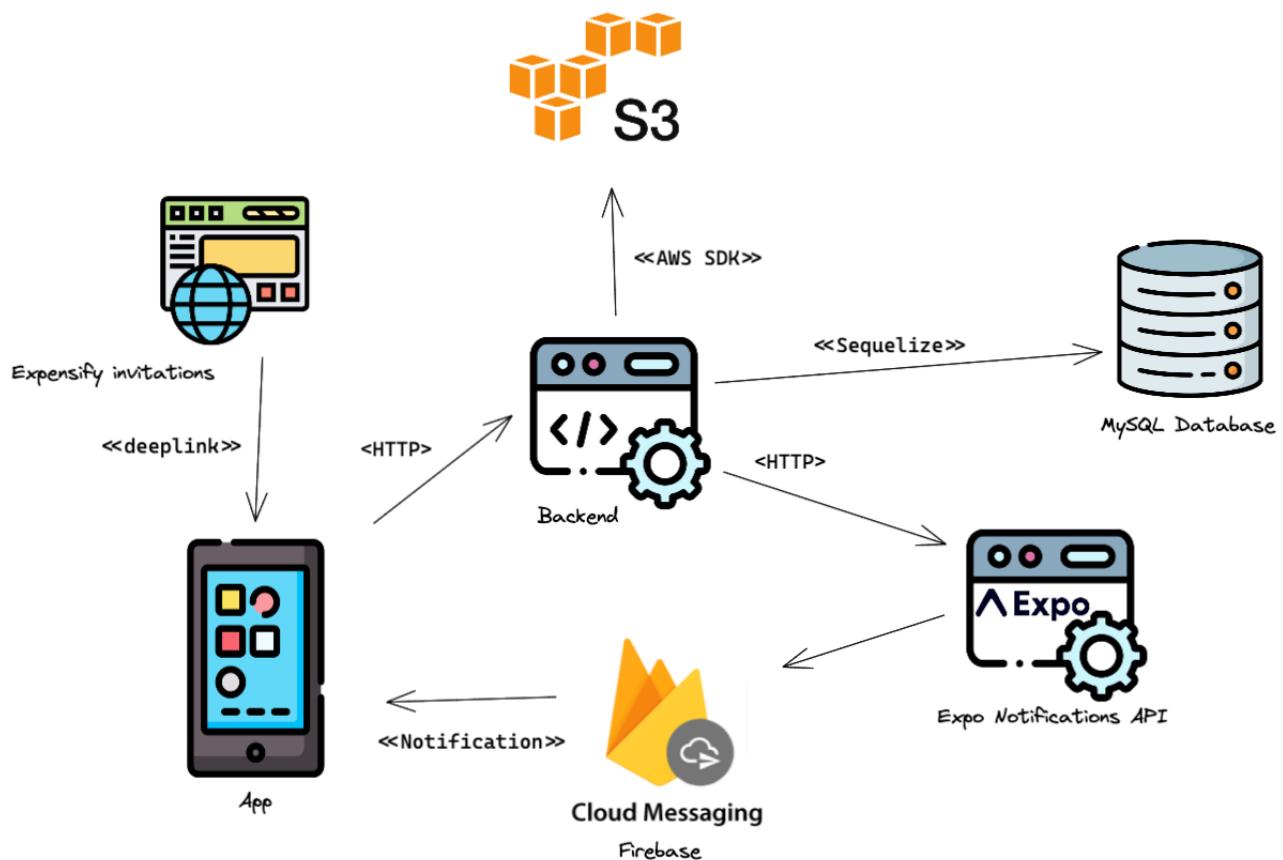
Se utilizará una estructura *package by layers*, lo que significa que cada capa tiene una responsabilidad. Este estilo permite testear fácilmente la aplicación (pudiendo probar cada capa por separado) y favorece la mantenibilidad.

Sobre el manejo de errores, se destaca el uso de la táctica ***Exception handling***. Cuando una excepción es lanzada, un *middleware* se encargará de renderizar la información del error en un mensaje que luego se le mostrará al usuario.

Todos los errores del sistema se encuentran agrupados en el paquete “errors” donde se guarda para cada uno un mensaje descriptivo y el *status code* del mismo.

Diseño de cada componente, mecanismos de comunicación, estructura de datos

Diagrama 3: Integraciones



Expensify invitations

Página web encargada de ser el primer punto con el que los usuarios se topan al abrir un link de una invitación, decidimos crearla para poder agregar el caso en que el usuario no tenga la aplicación instalada.

S3

Utilizado para almacenar las imágenes de Expenses y Categories.

App

Aplicación creada en react native.

Backend

Encargado de servir datos a la app.

MySQL Database:

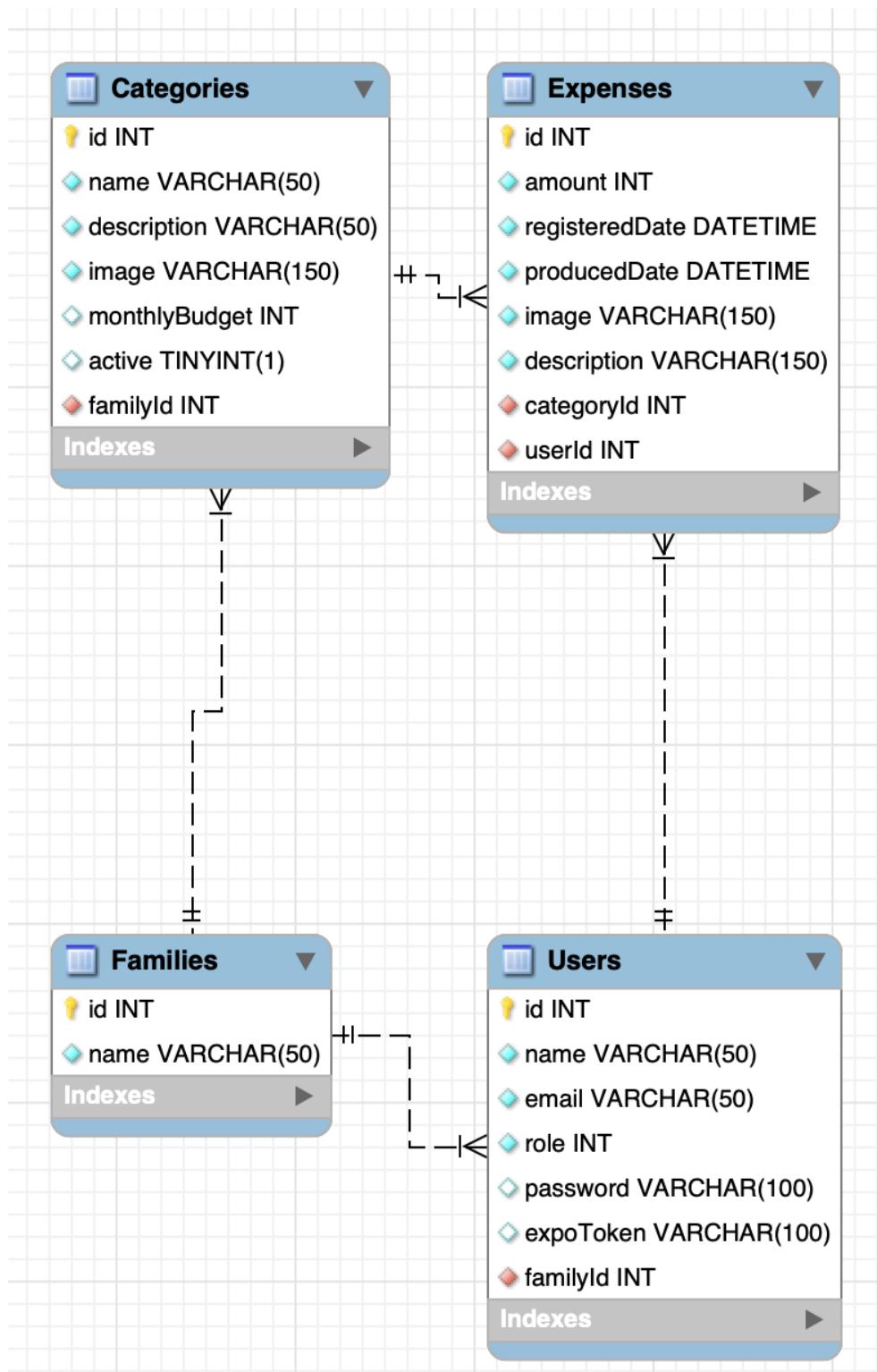
Encargada de almacenar todos los datos de la aplicación, exceptuando las imágenes.

Expo Notifications API y Firebase

Utilizados para el envío de notificaciones a los usuarios.

Modelo de tablas

Diagrama 4: Modelo de tablas



Justificaciones de la solución

Autenticación y autorización

Resistir ataques

Se utilizó la táctica **Authenticate actors** para validar la identidad de los usuarios. El sistema solicita un correo electrónico y contraseña del usuario, y en caso de que estas credenciales sean correctas, se le proporcionará un JWT (*Json Web Token*) que guarda la siguiente información: id del usuario, privilegios y el id de la familia a la que pertenece. Este token está firmado por la clave del servidor de la api, y si se identificara que no está bien formado, se le denegará el acceso al usuario informando del inconveniente. Cabe aclarar que el token mencionado se almacena en las cookies, teniendo estas una validez de un mes. Se decidió usar esta herramienta sobre el almacenamiento en *local* o *session storage*, ya que nos permite tener la seguridad de que el token no pueda ser accedido desde código en el frontend y ser enviado únicamente a través de el protocolo *https*. Encendiendo las flags *httpOnly: true* y *secure: true* respectivamente. Además todo el manejo se hace desde el backend con el header *Set-Cookie*, por lo cual no debemos preocuparnos por manejar el token de ninguna manera en el frontend.

Con el propósito de asegurar que un usuario tenga los permisos necesarios para acceder y/o modificar un recurso o servicio del sistema, se utilizó la táctica **Authorize actors**. Cuando se realiza una consulta al servidor, un middleware se encarga de llevar a cabo las tareas de control de acceso, en este caso basado en roles (siendo estos administrador y miembro).

A su vez, el token mencionado contiene el id de la familia de la cual el usuario forma parte, por lo cual, si este tuviese el propósito de consultar por los datos de otra familia, se le denegará el acceso a la información solicitada.

El sistema a su vez responde a URL mal formadas enviando al usuario a la página de inicio familiar si el usuario está logueado, o a la página de inicio de sesión en otro caso.

Recuperarse de ataques

Una vez que el sistema detectó e intentó de resistirse a un ataque, necesita recuperarse. Las tácticas de **Audit** y **Nonrepudiation** tuvieron lugar en este sistema. Se mantiene un historial de las acciones que realizan los usuarios, identificando a su vez el responsable de cada una. De

esta manera, si se detecta algo sospechoso, el usuario no puede negar los hechos y se puede encontrar la causa de problemas.

Seguridad

Para almacenar las contraseñas en la base de datos se genera un *salt* y este es luego *hasheado*. De esta manera, se elimina la posibilidad de un *Rainbow Table Attack* y se agrega una capa extra de seguridad a los datos sensibles.

Se cambió además la contraseña por defecto que tenía la base de datos a una más segura, favoreciendo así la táctica ***Change default settings***.

Por último, en el afán de proteger al sistema de ataques como *SQL Injection* (dónde código malicioso es insertado a través de consultas SQL), se manejó una ORM, particularmente sequelize. Esto lo que permite es evitar que se ejecute código no deseado en el sistema.

Portabilidad

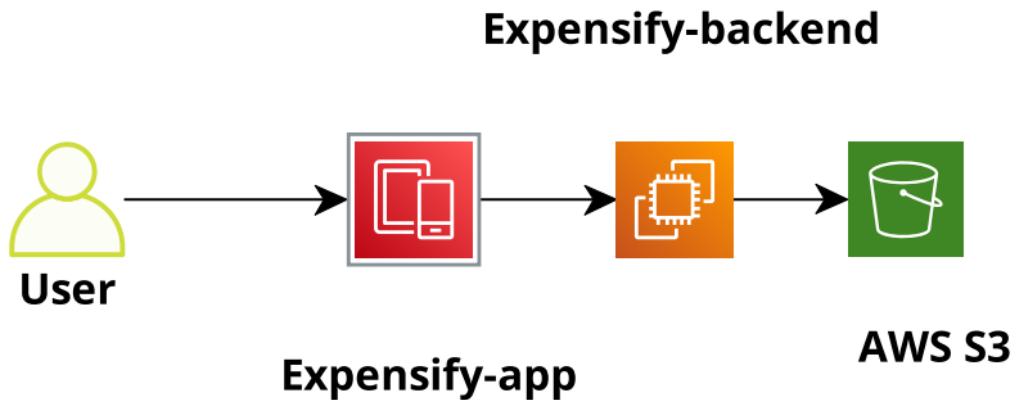
Los contenedores de Docker contienen todo lo necesario para ejecutar una aplicación, y por lo tanto, no es necesario descargarse dependencias de la aplicación directamente en el ordenador que la está corriendo. Estos contenedores se comparten fácilmente a través de imágenes, y quienes las reciban podrán hacer que funcione de la misma manera sin importar en dónde lo estén utilizando, lo que favorece la portabilidad.

Modificabilidad y desacoplamiento

Decidimos guardar las imágenes de las categorías y las expensas en AWS S3. El último se trata de un bucket que tiene como finalidad guardar archivos.

Por tanto, cuando se crea una expensa o una categoría, las imágenes son enviadas al backend y el último sube las imágenes al bucket correspondiente.

Diagrama 4: Envío de imágenes



En este sentido, el frontend no se acopla a la solución del backend, lo cual vela por la modificabilidad, ya que por si alguna razón el backend cambia, el frontend es ajeno a esto y no debe ser cambiado.

Por otro lado, tampoco se encuentra acoplado a lo que espera recibir el backend, ya que es el último el que resuelve esto.

Por supuesto, se trata de un trade-off, ya que enviar las imágenes al backend es costoso a nivel de performance. Sin embargo, esta solución vela por la modificabilidad, atributo que buscábamos maximizar.

Diseño de UI/UX

Nos guiamos en las heurísticas de Nielsen para diseñar la interfaz del usuario. Las imágenes de cada pantalla que evidencian la aplicación de las mismas se encuentran en el anexo.

Coincidencia entre el sistema y el mundo real

Se utilizaron palabras y conceptos familiares para los usuarios, y no términos orientados al sistema. De esta manera aumentamos la probabilidad de que los usuarios comprendan rápidamente lo que está pasando.

El usuario tiene control y libertad

El usuario puede navegar libremente y realizar las acciones que desee, teniendo también la posibilidad de deshacer esas acciones. Por ejemplo, un usuario podría crear una expensa y luego eliminarla, también podría cancelar la creación si se arrepiente mientras lo hace y también podría modificar las que ya están creadas.

Esta libertad le da al usuario la posibilidad de equivocarse sin presiones y sentir total control.

Consistencia y estándares

Se mantiene la consistencia a lo largo de todo el sistema.

- El color predominante es el fucsia en combinación con el blanco. El lenguaje es en inglés.
- Los botones de agregar y modificar son iguales en las expensas y en las categorías.
- Los mensajes de error están en rojo y los que indican el éxito en verde.
- Se mantiene el mismo formato en los formularios para crear/editar expensas y categorías.

Prevención de errores

Para cambios importantes, como lo es eliminar una expensa, se le pide una confirmación al usuario para que no realice un cambio que realmente no quiere hacer.

Reconocer en lugar de recordar

Se busca minimizar la carga de memoria del usuario haciendo visibles los objetos, acciones y opciones que tiene para realizar.

Estética y diseño minimalista

Se mantuvo un diseño minimalista, procurando no añadir más información de la estrictamente necesaria en cada pantalla. Los colores acompañan en esta estética para que no abrume al usuario mucho contenido visual.

Ayudar al usuario a reconocer, diagnosticar y recuperarse de los errores

Se muestra al usuario mensajes de error indicando claramente a qué se debe para que este pueda solucionarlo rápidamente.

Justificaciones de Clean Code

Nombres

Los nombres se pensaron para que sean descriptivos, pronunciables y fáciles de buscar. Por esta razón, se evita el uso de abreviaturas o de nombres que no revelan la utilidad de las variables/funciones.

A su vez, no se utilizan números mágicos, sino que se reemplaza a estos por constantes con nombre (por ejemplo, “nameLength” que guarda el máximo largo que puede tener el nombre de una categoría).

Mantener consistencia

Se apunta a la consistencia, por ejemplo, si el método para crear gastos se denomina “CreateExpense”, el de agregar categorías será “CreateCategory”. Además, la manera en que se agregan a sus respectivas tablas siguen la misma lógica siendo que son cosas similares.

Funciones pequeñas

Todas las funciones realizan una única cosa y se mantienen lo más pequeños posible.

Comentarios

Se estableció desde un principio intentar hacer que el código sea lo más legible posible y autoexplicativo. De esta manera, evita aclaraciones innecesarias que agregarían ruido a la solución.

Estructura del código fuente

El tamaño de las clases no exceden las 400 líneas de código y se establecen espacios en blanco para separar conceptos. Se mantienen las líneas bien indentadas, intentando también evitar alienación de código horizontal.

Pruebas

Pruebas independientes (su resultado no depende de otra prueba), con descripciones claras de lo que intentan probar. Se corren de manera muy rápida y sin importar cuántas veces se corra, siempre darán el mismo resultado.

Cobertura de pruebas

Ilustración 1: Cobertura de pruebas del 100% del código.

File	%Stmts	%Branch	%Funcs	%Lines	Uncovered Line #s
All files	100	100	100	100	
api/controllers	100	100	100	100	
categoryController.js	100	100	100	100	
expenseController.js	100	100	100	100	
familyController.js	100	100	100	100	
userController.js	100	100	100	100	
businessLogic	100	100	100	100	
categoryLogic.js	100	100	100	100	
expenseLogic.js	100	100	100	100	
familyLogic.js	100	100	100	100	
userLogic.js	100	100	100	100	
errors	100	100	100	100	
DuplicateCategoryError.js	100	100	100	100	
DuplicateFamilyError.js	100	100	100	100	
DuplicateUserError.js	100	100	100	100	
FileUploadError.js	100	100	100	100	
ForeignKeyError.js	100	100	100	100	
HttpRequestError.js	100	100	100	100	
ValidationError.js	100	100	100	100	
inputValidationrror.js	100	100	100	100	
errors/auth	100	100	100	100	
AuthError.js	100	100	100	100	
InviteTokenError.js	100	100	100	100	
library	100	100	100	100	
imageUploader.js	100	100	100	100	
jwtSupplier.js	100	100	100	100	
roles.js	100	100	100	100	
utilities	100	100	100	100	
dateUtils.js	100	100	100	100	
utilities/validators	100	100	100	100	
dateISOValidator.js	100	100	100	100	
emailValidator.js	100	100	100	100	
inArrayValidator.js	100	100	100	100	
numberValidator.js	100	100	100	100	
paragraphValidator.js	100	100	100	100	
passwordValidator.js	100	100	100	100	
wordValidator.js	100	100	100	100	


```
Test Suites: 12 passed, 12 total
Tests:      135 passed, 135 total
Snapshots:  0 total
Time:       6.558 s, estimated 8 s
```

Se cuenta con pruebas unitarias de todas las funcionalidades del sistema que se corren automáticamente cada vez que se realiza un pull request a la rama develop, favoreciendo así la integración continua.

Retrospectiva general

En términos generales, este proyecto fue muy enriquecedor ya que nos topamos con un paradigma con el que no habíamos tenido experiencia antes en la carrera. Por lo tanto, nos enfrentamos a diversos desafíos resultantes de una tecnología no utilizada anteriormente , sumergiéndonos en un mundo desconocido como es el desarrollo móvil.Esto nos permitió tanto conocer nuevas tecnologías como adaptarnos a circunstancias que desconocemos.

No obstante, estos desafíos fueron afrontados a lo largo del obligatorio; elementos tales como documentación y videos fueron de gran utilidad a la hora de poder resolverlos. Asimismo, la colaboración entre las partes fue importante cuando había algo que nos bloqueaba.

En términos de lo logrado, estamos satisfechos ya que pudimos cumplir con todos los requerimientos funcionales que nos habíamos plantado en el Planning y además creemos que llegamos a un buen resultado.

Consideramos que fue importante también plantearnos desafíos como el uso de la cámara, notificaciones, *deeplinks*, interacción con otras aplicaciones que si bien aumentaron el riesgo de poder completar la aplicación, fueron muy beneficiosos para el aprendizaje.

Instructivo de instalación

Pre requisitos:

- Docker
- Node
- JDK
- Android Studio
- [Ambiente de React Native configurado correctamente](#)

Backend

```
git clone https://github.com/ArqSoftPractica/Monolithic-Backend
cd Monolithic-Backend
npm i
docker-compose up
```

App

Develop:

```
yarn install
yarn android
```

Build apk:

```
eas build -p android --profile preview --local
```

Nota:

También se puede encontrar un apk para instalar en la carpeta release.

Aplicación de principios arquitectónicos específicos de React Native

Componentes

Se decidió utilizar componentes funcionales ya que es el estándar de la tecnología y nos habilita a usar otras herramientas como los hooks, que generalmente son la forma de interactuar con la mayoría de las librerías y se utilizan para conceptos centrales como State, Ref, Effect, etc

Navegación

Para la navegación decidimos usar un stack navigator, ya que es una de las opciones más sólidas para aplicaciones con un flujo simple como lo es nuestro caso.

Una ventaja importante de '@react-navigation/stack', fue que nos permitió implementar de forma muy sencilla los deeplinks, ya que nos crea automáticamente todos los manejadores para las respectivas pantallas y al usar el link nos redirige a ellas.

Ejemplo: `expensify://screens/register/{params}`

Se creó la navegación pensando en dos ramas principales:

- Flujo normal:
 - Utilizado para la navegación de la app una vez el usuario está logueado.
 - Pantallas:
 - Home
 - ExpenseDetails
 - ExpenseForm
 - Categories
 - CategoryDetails
 - CategoryForm
 - Analysis
 - Configuration

- Autenticacion
 - Utilizado para la navegación de la app cuando el usuario no está logueado.
 - Pantallas:
 - Register
 - SignIn

Hooks

Decidimos apoyarnos en hooks tales como useState, useEffect, useContext para el manejo de la UI y el estado de la app. Además creamos hooks propios para poder reutilizar funciones en diferentes pantallas (Ej: useQueryAuth).

Context

Para el manejo de estados compartido por diferentes pantallas se decidió usar Context, de esta manera podemos acceder a los datos entre pantallas sin tener que mandarlos como parámetros de ruta.

Se crearon dos Contexts:

- AuthContext:
 - Encargado de manejar la sesión del usuario y almacenar los datos relevantes para la misma y de esta manera poder persistirla.
- AlertContext:
 - Encargado de manejar los mensajes de error y éxito en toda la app.

Queries

Para el manejo de requests al backend se decidió usar useQuery, ya que provee varios hooks que facilitan el manejo de la información de las mismas, como lo son useQuery y useInfiniteQuery. Este último nos permitió reducir el overhead a la hora de fetchear listas, ya que nos facilitó la implementación del paginado, evitando traer todos los datos del backend.

Otro hook a destacar fue useMutation, que utilizamos para las request de POST/PUT/DELETE, que facilitó el manejo caso de éxito y error.

Anexo

Anexo 1: Historias de usuario

US1: Registro de usuario administrador
<p><i>Actor:</i> Usuario administrador</p>
<p><i>Narrativa:</i></p> <p>Como usuario administrador Quiero poder registrarme a mí y a mi familia en la aplicación Para poder tener un control de los gastos de mi familia.</p>
<p><i>Criterios de aceptación:</i></p> <p>Escenario 1: Crear usuario con nombre inválido Dado un nombre inválido Cuando el usuario quiere registrarse Entonces debe recibir un mensaje de error</p> <p>Escenario 2: Crear usuario con mail inválido Dado un mail con dominio inválido Cuando el usuario quiere registrarse Entonces debe recibir un mensaje de error</p> <p>Escenario 3: Crear usuario con nombre de familia ya ingresado Dado un nombre de familia ya ingresado en el sistema Cuando el usuario quiere registrarse Entonces debe recibir un mensaje de error</p> <p>Escenario 4: Crear usuario con contraseña invalida Dado una contraseña inválida Cuando el usuario quiere registrarse Entonces debe recibir un mensaje de error</p> <p>Escenario 5: Crear usuario con datos válido Dado un nombre de familia , nombre, email, contraseña válida Cuando el usuario quiere registrarse Entonces mi usuario y mi familia se crean y soy redirigido a la pantalla principal</p>
<p>Nota: Se considera nombre válido a un nombre con máximo 20 caracteres, conteniendo solamente letras, números y/o espacios. Se considera contraseña válida a una contraseña de entre 8 y 64 caracteres, conteniendo únicamente letras, números y/o símbolos.</p>

US2: Enviar invitación

Actor: Usuario administrador

Narrativa:

Como usuario administrador

Quiero poder enviar una invitación a través de Whatsapp o Email

Para poder agregar a otros miembros y administradores a la misma

Criterios de aceptación:

Escenario 1: Crear invitación

Dado usuario administrador logueado

Cuando el usuario presiona enviar invitación

Entonces se crea la invitación y se le dan las opciones de enviar por Whatsapp o email.

US3: Aceptar invitación

Actor: Usuario administrador y usuario miembro

Narrativa:

Como usuario

Quiero poder aceptar una invitación a una familia

Para poder tener acceso a los gastos de la misma

Criterios de aceptación:

Escenario 1: Aceptar invitación sin estar logueado

Dado usuario no logueado

Cuando el usuario abre el link de invitación

Entonces se abre la aplicación en la pantalla de registro con el rol que se creará el usuario y a la familia que pertenecerá

Escenario 2: Aceptar invitación estando logueado

Dado usuario logueado

Cuando el usuario abre el link de invitación

Entonces se abre la aplicación mostrando un mensaje de error

Nota:

Tanto los usuarios administradores como miembros sólo podrán pertenecer a una única familia.

US4: Iniciar sesión

Actor: Usuario administrador y usuario miembro

Narrativa:

Como usuario

Quiero ingresar mis credenciales (email y contraseña)

Para poder acceder a la aplicación

Criterios de aceptación:

Escenario 1: Inicio de sesión con email inválido

Dado un email con dominio invalido

Cuando presiona el botón de ingresar

Entonces se muestra un mensaje de error

Escenario 2: Inicio de sesión con email y contraseña incorrectas

Dado un email y contraseña que no coinciden

Cuando presiona el botón de ingresar

Entonces se muestra un mensaje de error

Escenario 3: Inicio de sesión con email y contraseña correctas

Dado un email y contraseña que coinciden

Cuando presiona el botón de ingresar

Entonces se redirige a la página principal

US5: Cerrar sesión

Actor: Usuario administrador y usuario miembro

Narrativa:

Como usuario logueado

Quiero desloguearme de la aplicación

Para poder acceder al inicio de sesión y registro

Criterios de aceptación:

Escenario 1: Cierre de sesión

Dado un usuario logueado

Cuando presiona el botón de cerrar sesión

Entonces se redirige a la página de inicio de sesión

US6: Ver categorías

Actor: Usuario administrador

Narrativa:

Como usuario administrador
Quiero poder ver la lista categorías
Para saber cuales están ya creadas

Criterios de aceptación:

Escenario 1: Ver categorías correctamente
Dada una lista de categorías
Cuando el usuario quiere ver las categorías
Entonces se listan las categorías

US7: Agregar categoría

Actor: Usuario administrador

Narrativa:

Como usuario administrador
Quiero poder agregar categorías
Para poder clasificar los gastos según estas.

Criterios de aceptación:

Escenario 1: Agregar categoría correcta
Dada una categoría con datos correctos
Cuando el usuario intenta agregar la categoría
Entonces debería agregarse a la lista de categorías

Escenario 2: Agregar categoría incorrecta
Dada una categoría con datos incorrectos
Cuando el usuario intenta agregar la categoría
Entonces se muestra un mensaje de error al usuario

Nota:

Una categoría se considera correcta cuando

- El nombre es alfanumérico no vacío de hasta 20 caracteres
- La descripción es alfanumérica no vacía de hasta 20 caracteres
- Que haya una imagen seleccionada
- El presupuesto mensual (opcional) es mayor a 0

US8: Modificar categoría

Actor: Usuario administrador

Narrativa:

Como usuario administrador

Quiero poder modificar categorías

Para poder cambiar datos de ésta en caso de necesitarlo.

Criterios de aceptación:

Escenario 1: Modificar nombre de categoría correctamente

Dada una categoría existente y un nombre válido

Cuando el usuario intenta modificar el nombre de la categoría con el nuevo nombre

Entonces el nombre de la categoría debería verse modificado

Escenario 2: Modificar descripción de categoría correctamente

Dada una categoría existente y una descripción válida

Cuando el usuario intenta modificar la descripción de la categoría con la nueva descripción

Entonces la descripción de la categoría debería verse modificada

Escenario 3: Modificar imagen de categoría correctamente

Dada una categoría existente y una imagen válida

Cuando el usuario intenta modificar la imagen de la categoría con la nueva imagen

Entonces la imagen de la categoría debería verse modificada

Escenario 4: Modificar presupuesto mensual de categoría correctamente

Dada una categoría existente y un presupuesto mensual válido

Cuando el usuario intenta modificar el presupuesto mensual de la categoría con el nuevo presupuesto mensual

Entonces el presupuesto de la categoría debería verse modificado

Escenario 5: Modificar categoría incorrecta

Dada una categoría y datos incorrectos

Cuando el usuario intenta modificar la categoría con datos incorrectos

Entonces se muestra un mensaje de error al usuario

Nota:

Una categoría se considera correcta cuando

- El nombre es alfanumérico no vacío de hasta 20 caracteres
- La descripción es alfanumérica no vacía de hasta 20 caracteres
- Que haya una imagen seleccionada
- El presupuesto mensual (opcional) es mayor a 0

US9: Eliminar categoría

Actor: Usuario administrador

Narrativa:

Como usuario administrador
Quiero poder eliminar categorías
Para no tener categorías que no necesito.

Criterios de aceptación:

Escenario 1: Eliminar categoría correctamente
Dada una categoría existente
Cuando el usuario intenta eliminar la categoría
Entonces debería eliminarse de la lista de categorías

US10: Ver alerta por superar presupuesto

Actor: Usuario administrador

Narrativa:

Como usuario administrador
Quiero recibir alertas cuando se exceda el presupuesto mensual de una categoría
Para tener un control sobre los gastos

Criterios de aceptación:

Escenario 1: Recibir alerta
Dados gastos de una misma categoría
Cuando la cantidad gastada supera el presupuesto mensual de la categoría
Entonces debería llegar una notificación al administrador

US11: Agregar gasto

Actor: Usuario administrador y miembro

Narrativa:

Como usuario
Quiero poder agregar gastos
Para poder tener registro de estos.

Criterios de aceptación:

Escenario 1: Crear usuario con nombre inválido
Dado un gasto con datos correctos
Cuando el usuario intenta agregar el gasto
Entonces debería agregarse a la lista de gastos

Escenario 2: Crear usuario con mail inválido
Dado un gasto con datos incorrectos
Cuando el usuario intenta agregar el gasto
Entonces se muestra un mensaje de error

Nota: un gasto se considera correcta cuando
La cantidad gastada es un número mayor a 0, con un máximo de 100.000.000
Que haya una fecha seleccionada
Que haya una categoría seleccionada

US12: Modificar de gasto

Actor: Usuario administrador

Narrativa:

Como usuario administrador

Quiero poder modificar gastos

Para poder cambiar datos de este en caso de que estén erróneos.

Criterios de aceptación:

Escenario 1: Modificar cantidad de gasto correctamente

Dado un gasto existente y una cantidad válida

Cuando el usuario intenta modificar la cantidad del gasto con la nueva cantidad

Entonces la cantidad del gasto debería verse modificada

Escenario 2: Modificar fecha de producido de gasto correctamente

Dado un gasto existente y una fecha válida

Cuando el usuario intenta modificar la fecha de producido del gasto con la nueva fecha

Entonces la fecha del producido del gasto debería verse modificada

Escenario 3: Modificar categoría del gasto correctamente

Dado un gasto existente y una categoría existente

Cuando el usuario intenta modificar la categoría del gasto con la nueva categoría

Entonces la categoría del gasto debería verse modificada

Escenario 4: Crear usuario con contraseña invalida

Dado una contraseña inválida

Cuando el usuario quiere registrarse

Entonces debe recibir un mensaje de error

Escenario 5: Crear usuario con datos válido

Dado un nombre de familia , nombre, email, contraseña válida

Cuando el usuario quiere registrarse

Entonces mi usuario y mi familia se crean y soy redirigido a la pantalla principal

Nota: un gasto se considera correcta cuando

La cantidad gastada es un número mayor a 0, con un máximo de 100.000.000

Que haya una fecha seleccionada

Que haya una categoría seleccionada

US13: Eliminar gasto

Actor: Usuario administrador

Narrativa:

Como usuario administrador
Quiero poder eliminar categorías
Para no tener categorías que no necesito.

Criterios de aceptación:

Escenario 1: Eliminar categoría correctamente

Dada una categoría existente
Cuando el usuario intenta eliminar la categoría
Entonces debería eliminarse de la lista de categorías

US14: Pagina de inicio familiar

Actor: Usuario administrador y miembro

Narrativa:

Como usuario administrador o miembro
Quiero poder ver en la pantalla principal los gastos registrados en lo que va del mes actual
Para poder ver cuánto voy gastando en el mes y para qué categoría

Criterios de aceptación:

Escenario 1: Visualizar página de inicio familiar
Dado que soy un usuario administrador o miembro
Cuando inicio sesión
Entonces debo poder ver cuánto voy gastando en el mes y para qué categoría

Nota:

Se considera nombre válido a un nombre con máximo 20 caracteres, conteniendo solamente letras, números y/o espacios.

Se considera contraseña válida a una contraseña de entre 8 y 64 caracteres, conteniendo únicamente letras, números y/o símbolos.

US15: Análisis de gastos por periodo

Actor: Usuario administrador

Narrativa:

Como usuario administrador o miembro

Quiero poder filtrar por período los gastos

Para poder ver cuánto gasté en ese período, separado por semanas o meses

Criterios de aceptación:

Escenario 1: Visualizar gastos por meses

Dado que soy un usuario administrador o miembro

Cuando inicio sesión

Entonces debo poder elegir un período dado (al menos un mes) y ver cuánto gasté en esos meses.

Escenario 1: Visualizar gastos por semana

Dado que soy un usuario administrador o miembro

Cuando inicio sesión

Entonces debo poder elegir un período dado (menos de un mes) y ver cuánto gasté en esos meses.

US16: Análisis de gastos por categoría

Actor: Usuario administrador

Narrativa:

Como usuario administrador o miembro

Quiero poder ver en un gráfico los gastos para el período dado agrupados por categoría

Para poder ver cuánto gasté en ese período y para qué categoría

Criterios de aceptación:

Escenario 1: Ver gráfico

Dado que inicié sesión y seleccioné un período dado

Cuando ingresó en análisis de datos

Entonces debería ver los gastos para el período seleccionado agrupados por categorías

Anexo 2: Prototipo

Ilustración 2: Registro de usuario administrador

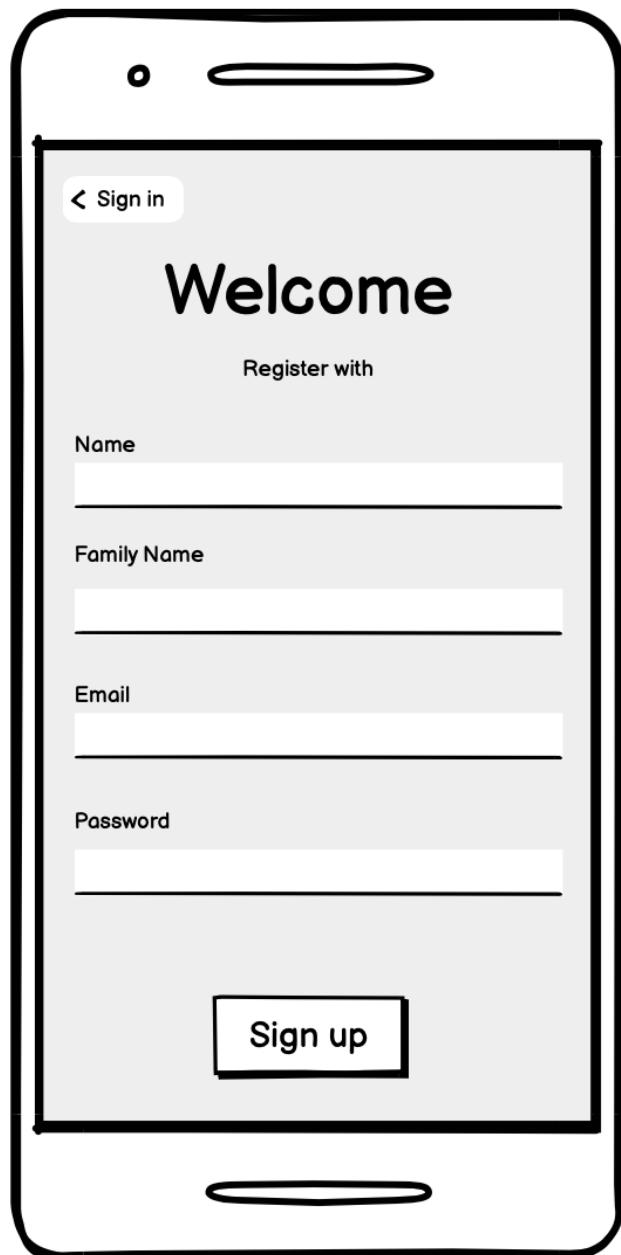


Ilustración 3: Registro de usuarios mediante invitación

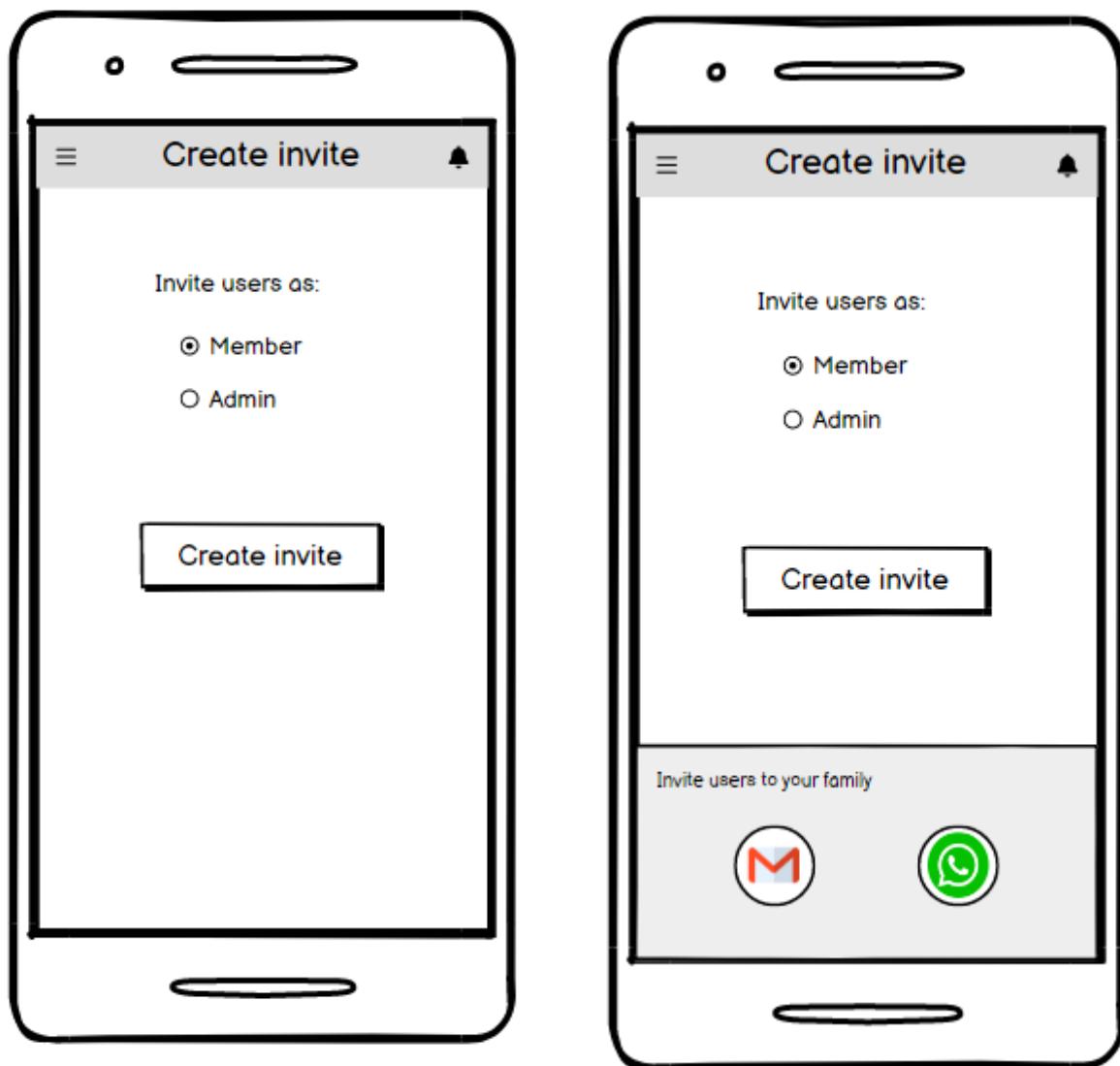


Ilustración 4: Autenticación de usuario



Ilustración 5: Pantalla de categorías y detalle de categoría

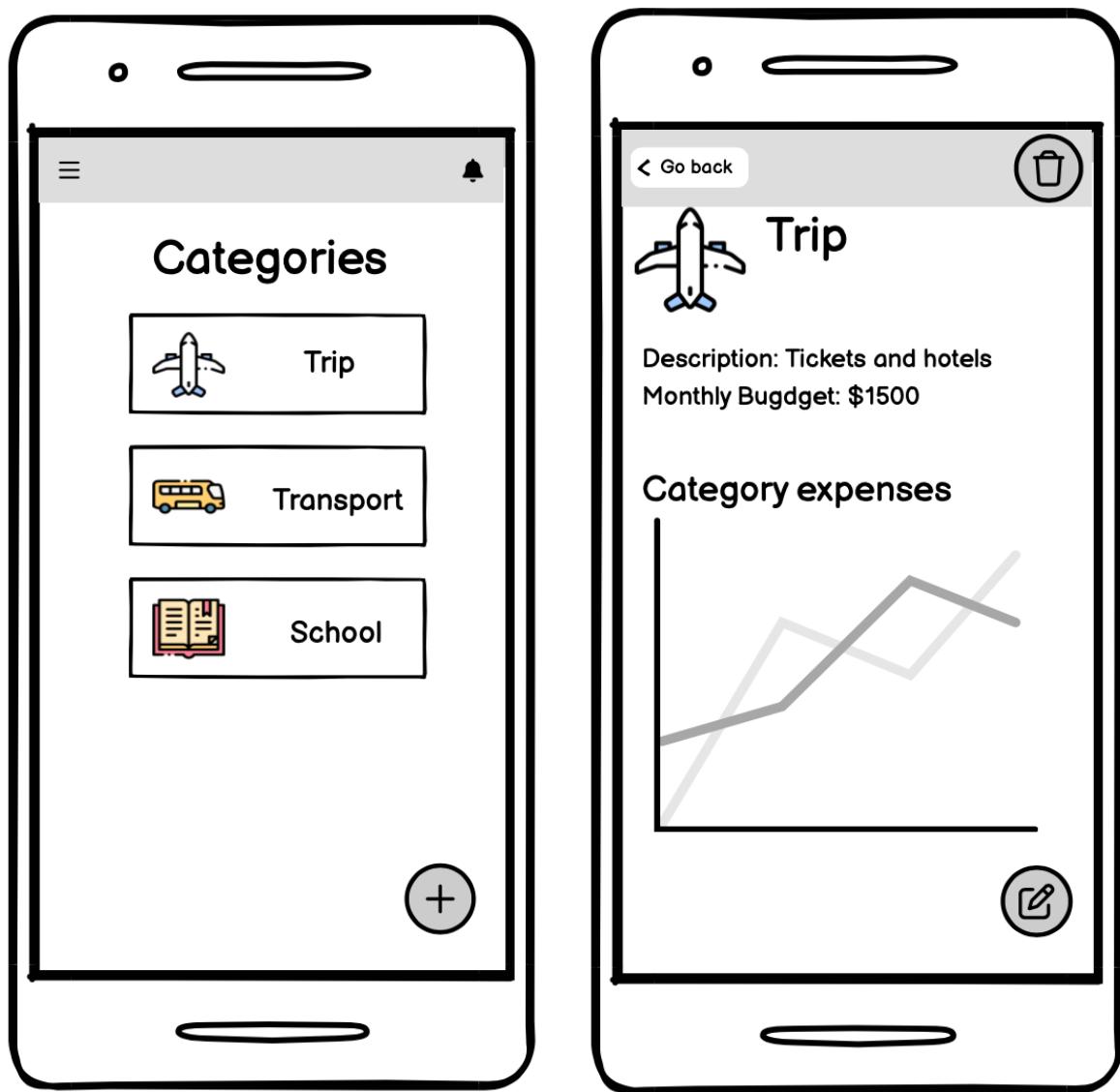


Ilustración 6: Agregar categoría

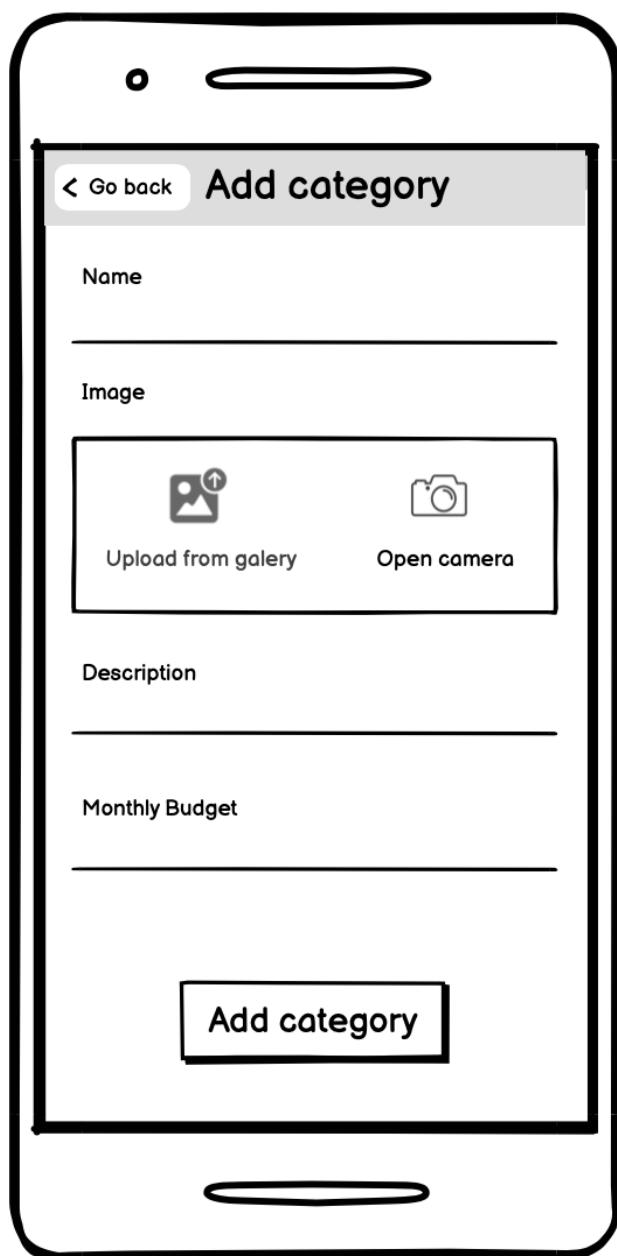


Ilustración 7: Página de inicio familiar

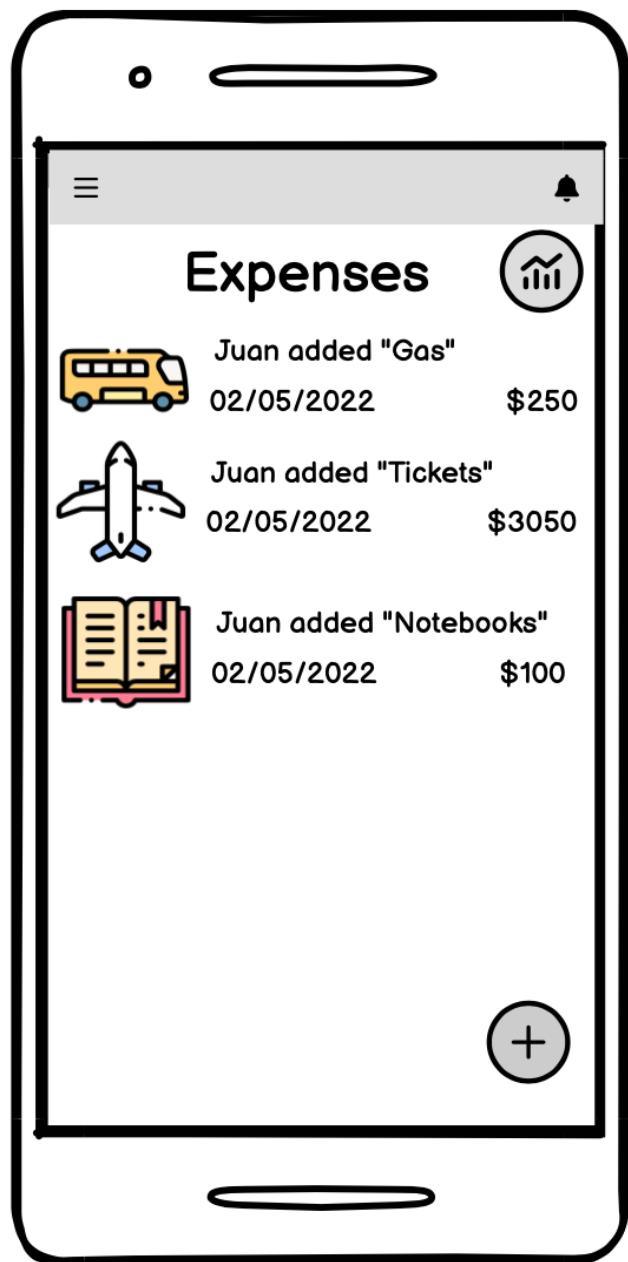


Ilustración 8: Modificar categoría

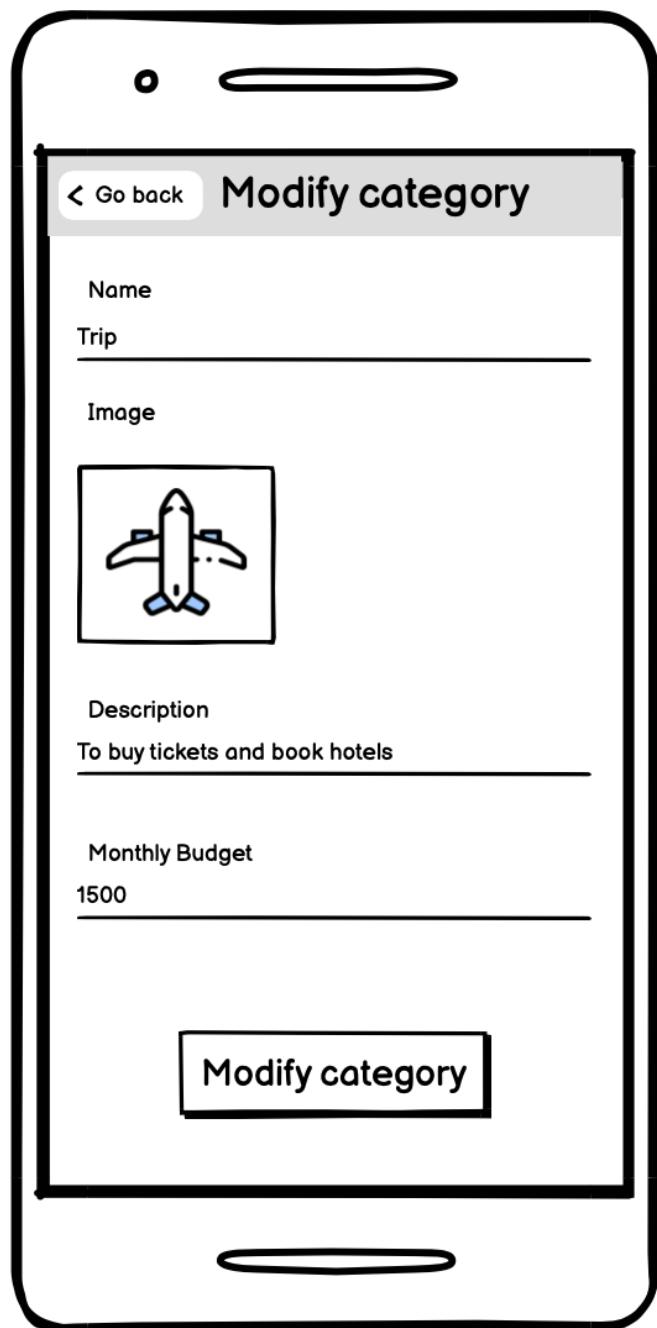


Ilustración 9: Eliminar categoría

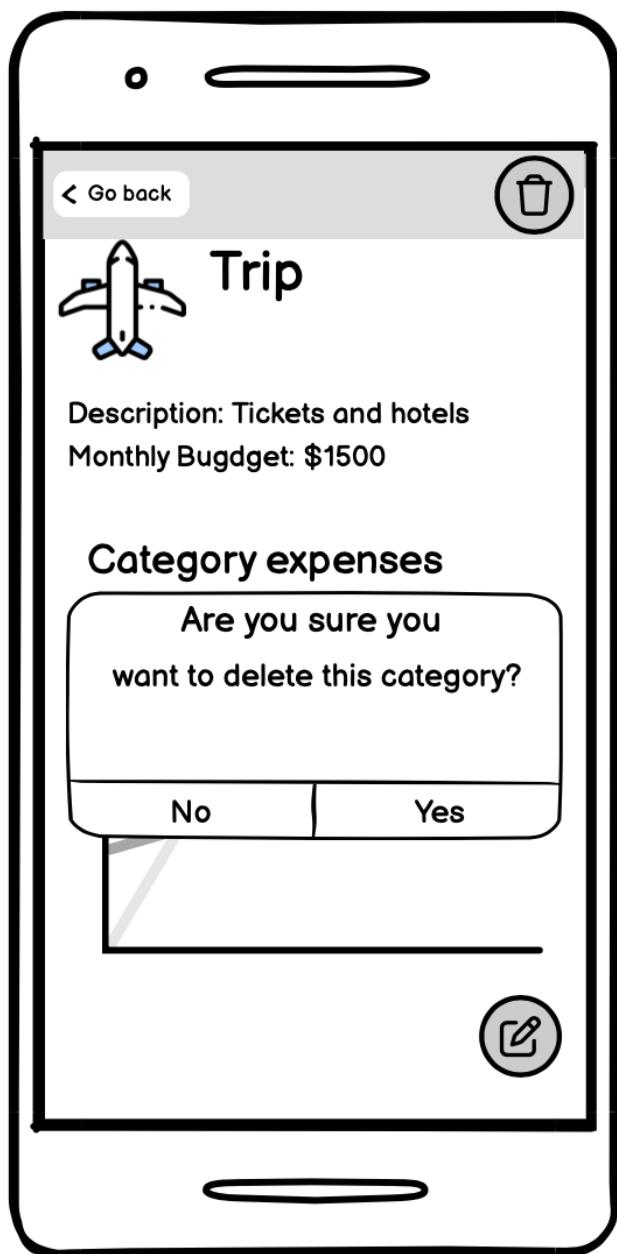


Ilustración 10: Alerta por superación de presupuesto mensual

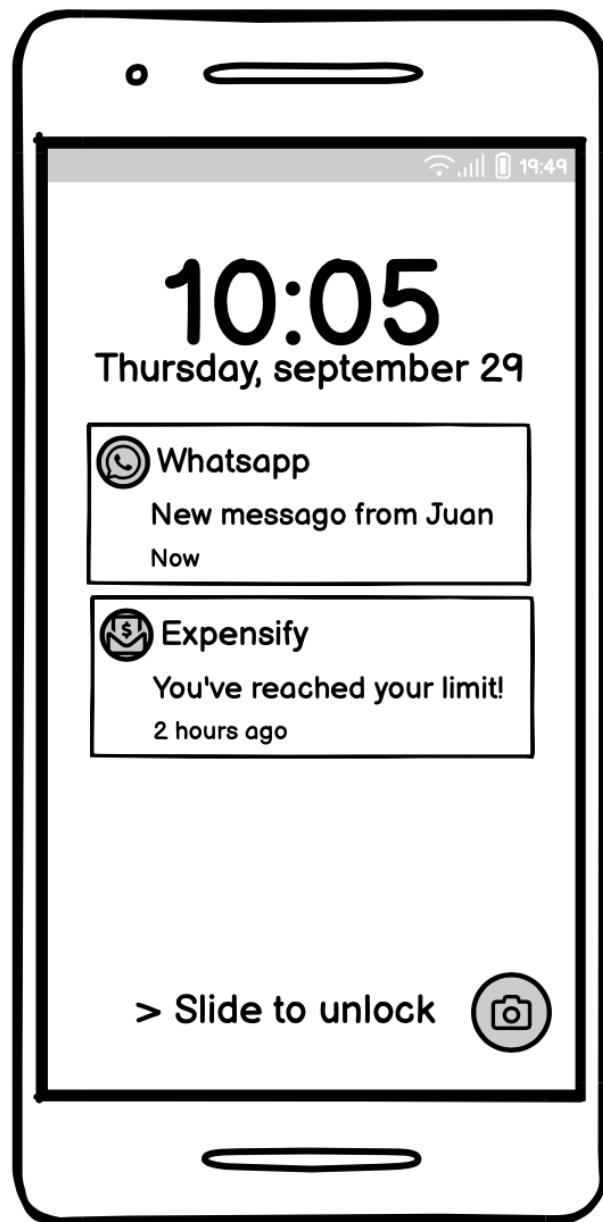


Ilustración 11: Agregar gasto

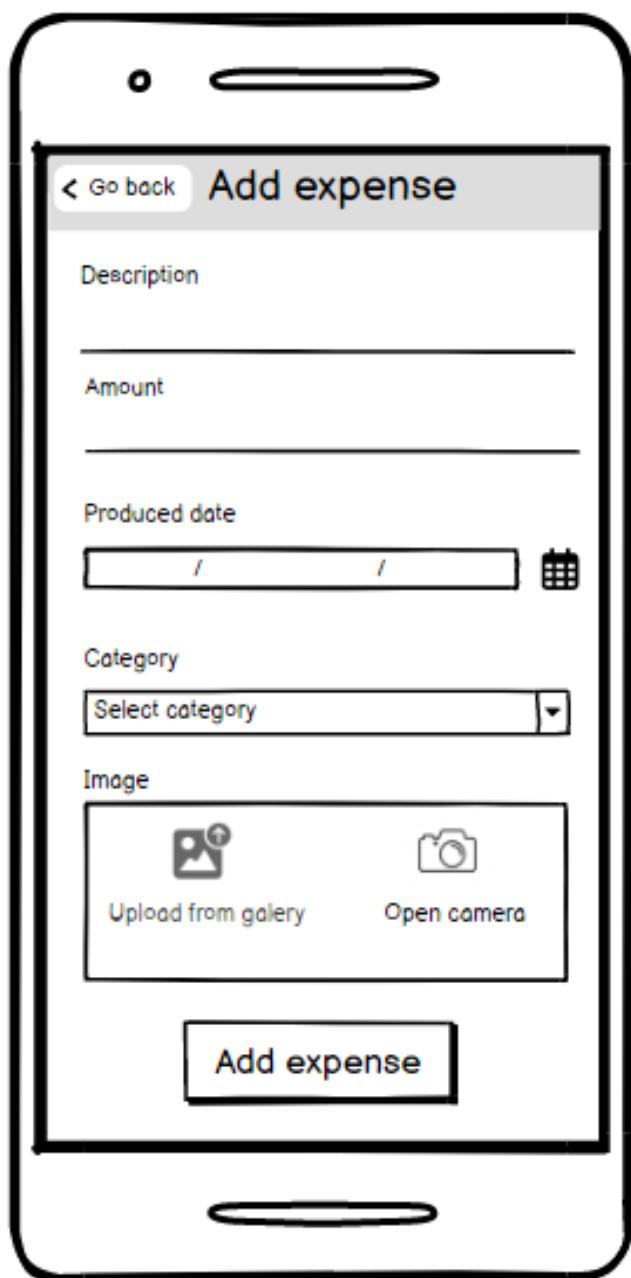


Ilustración 12: Modificar gasto

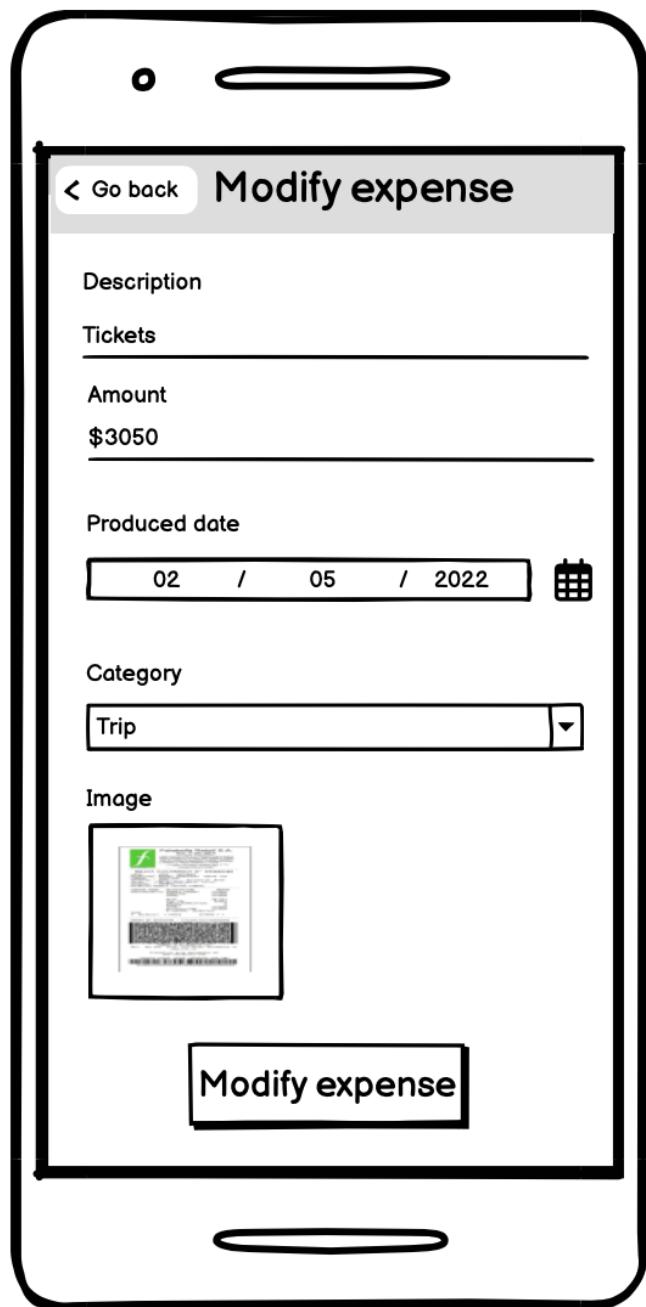


Ilustración 13: Eliminar gasto

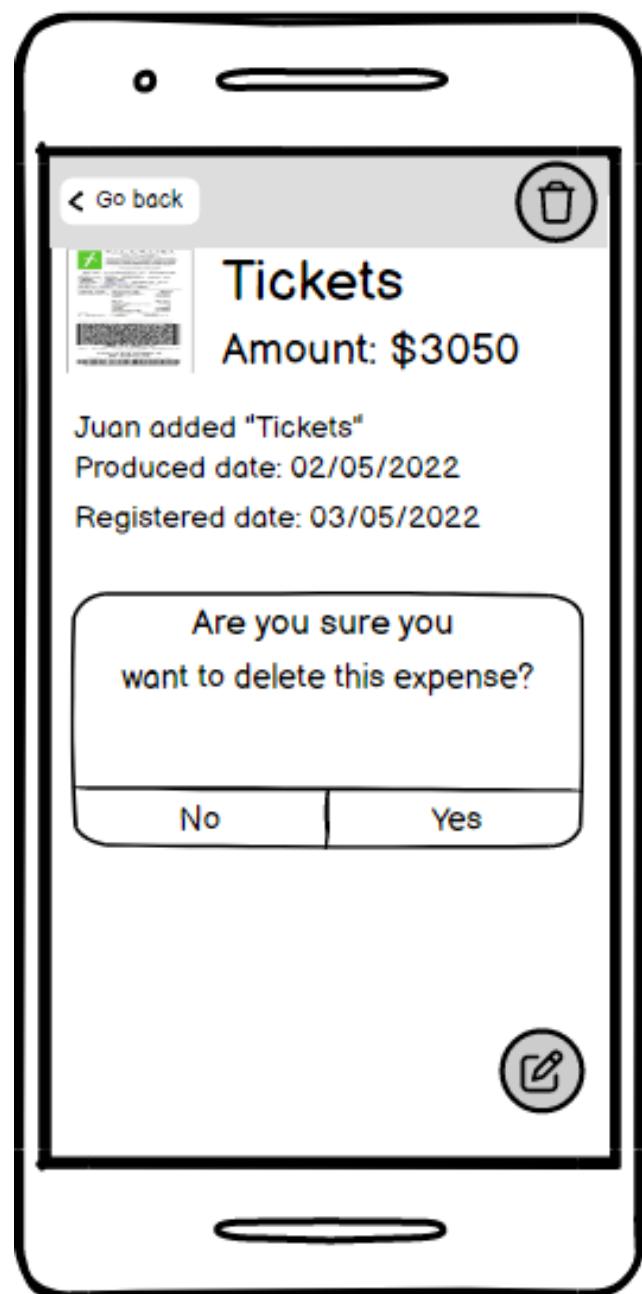


Ilustración 14: Página de inicio familiar

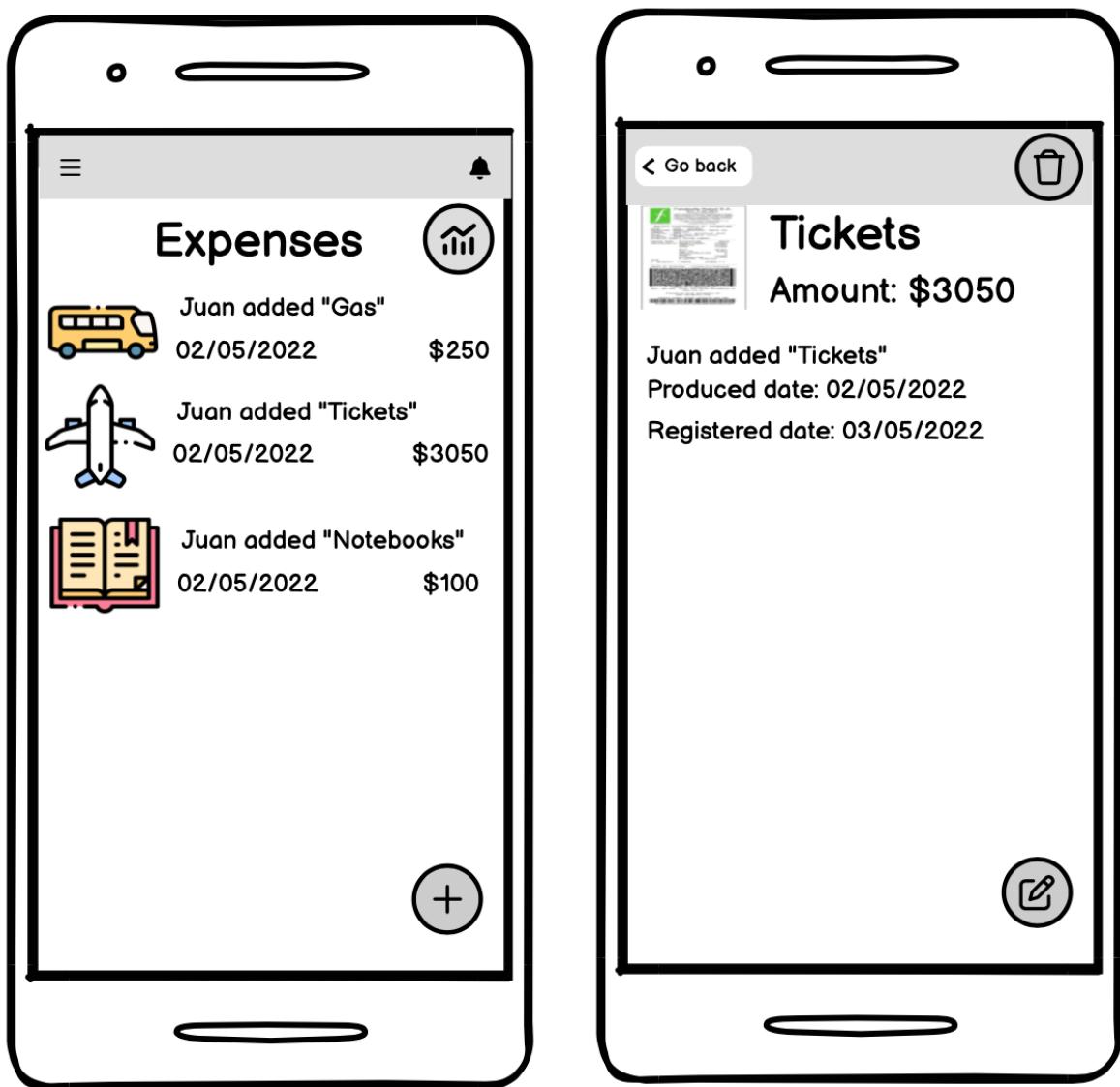
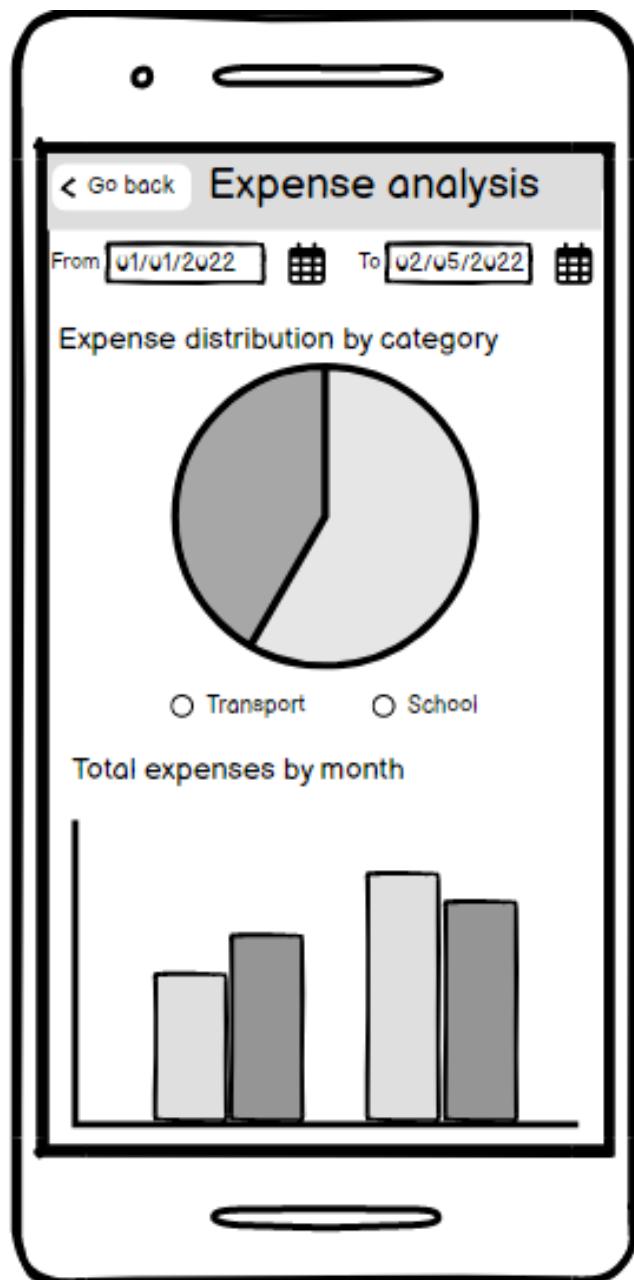


Ilustración 15: Análisis de gastos



Anexo 3: Aplicación

Ilustración 16: Ícono de la aplicación

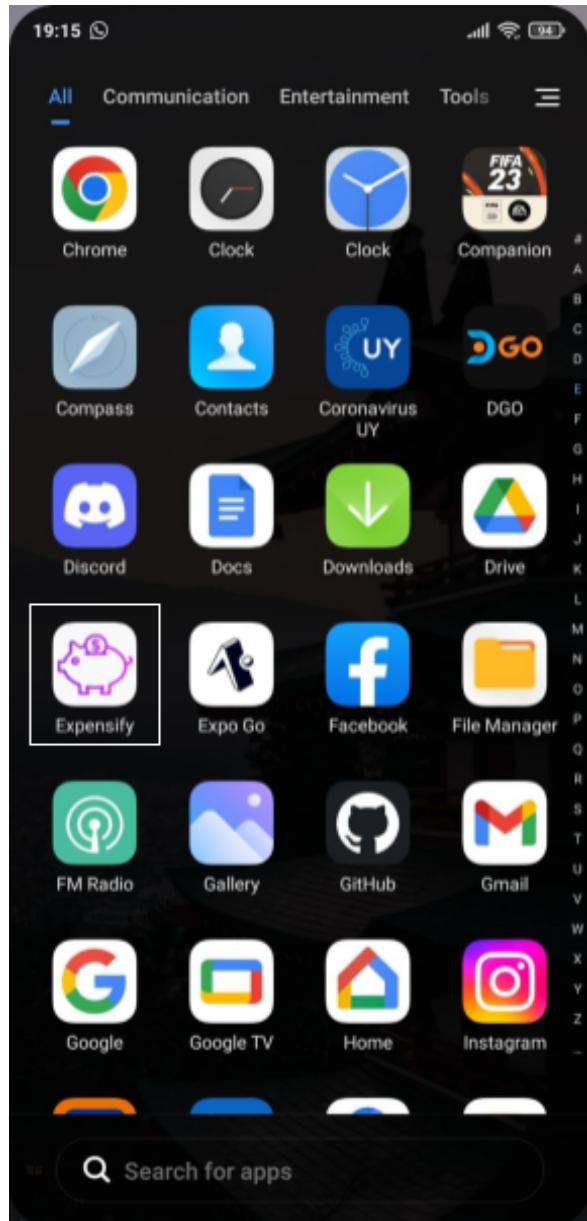


Ilustración 17: Splash screen



Ilustración 18: Registro de usuario administrador

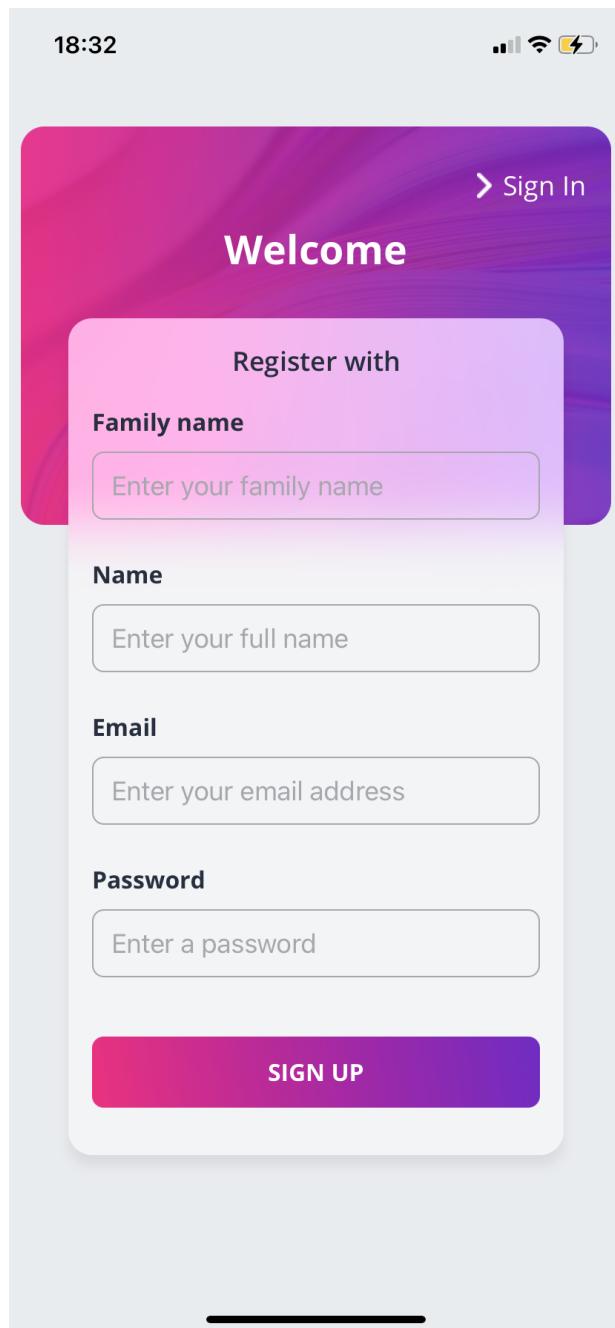


Ilustración 19: Registro de usuario mediante invitación

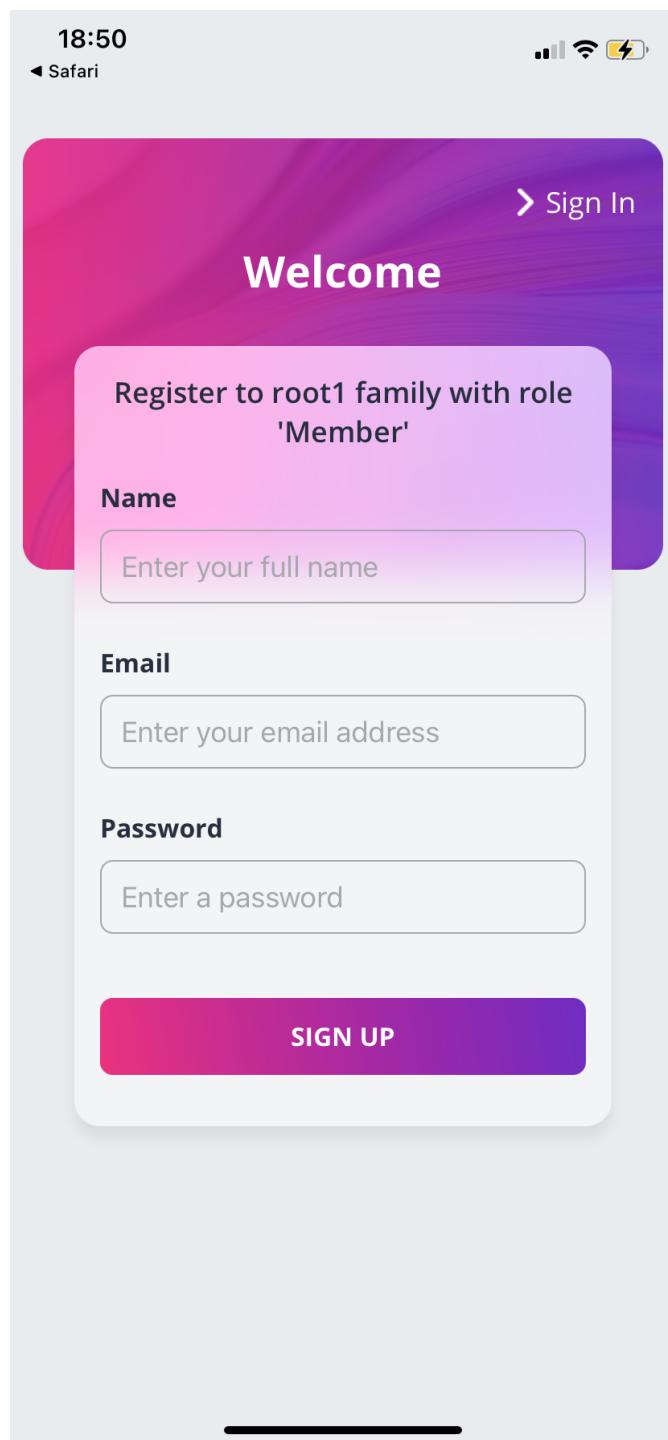


Ilustración 20: Autenticación de usuario

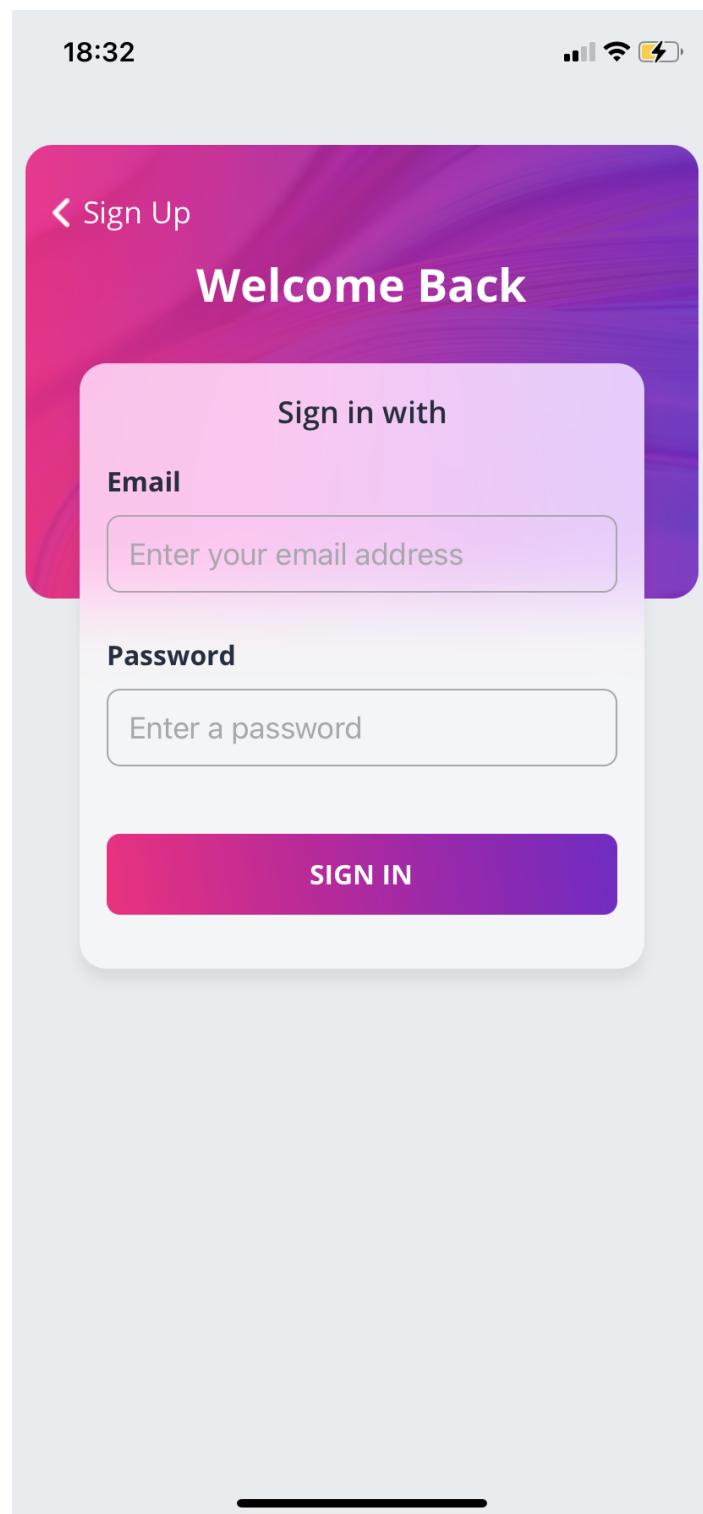


Ilustración 21: Menú lateral

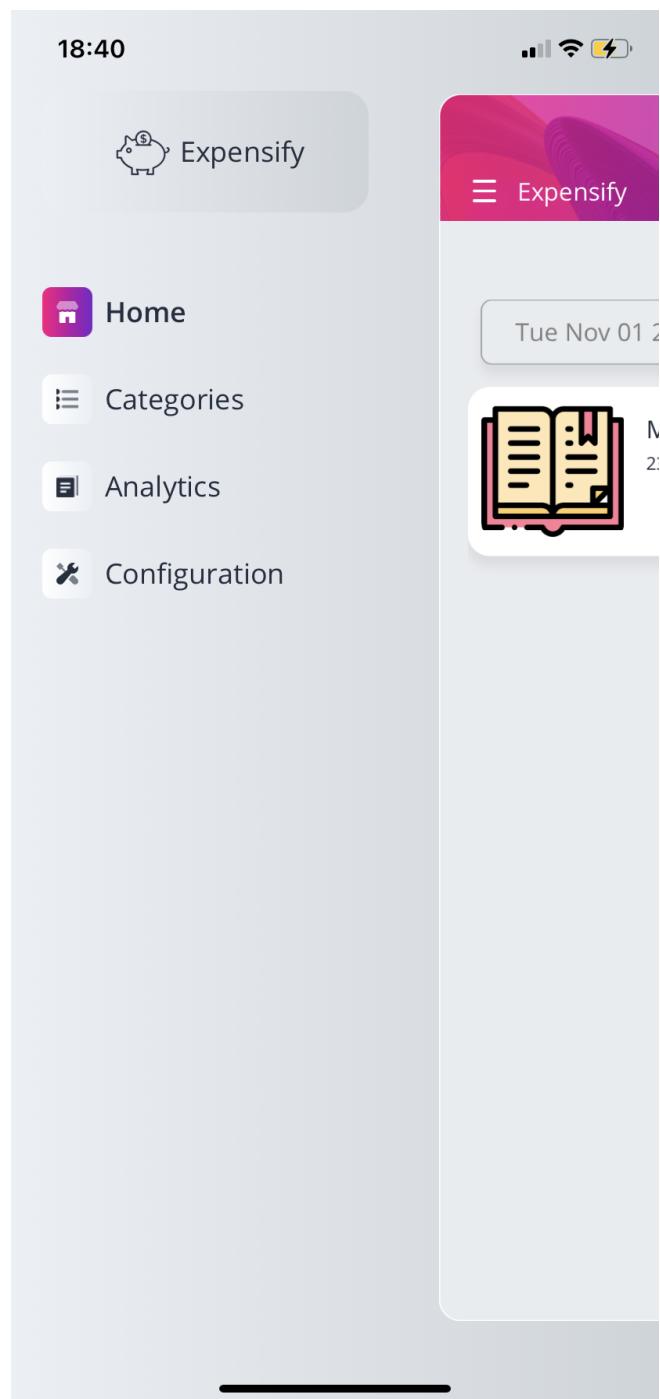


Ilustración 22: Pantalla de inicio familiar

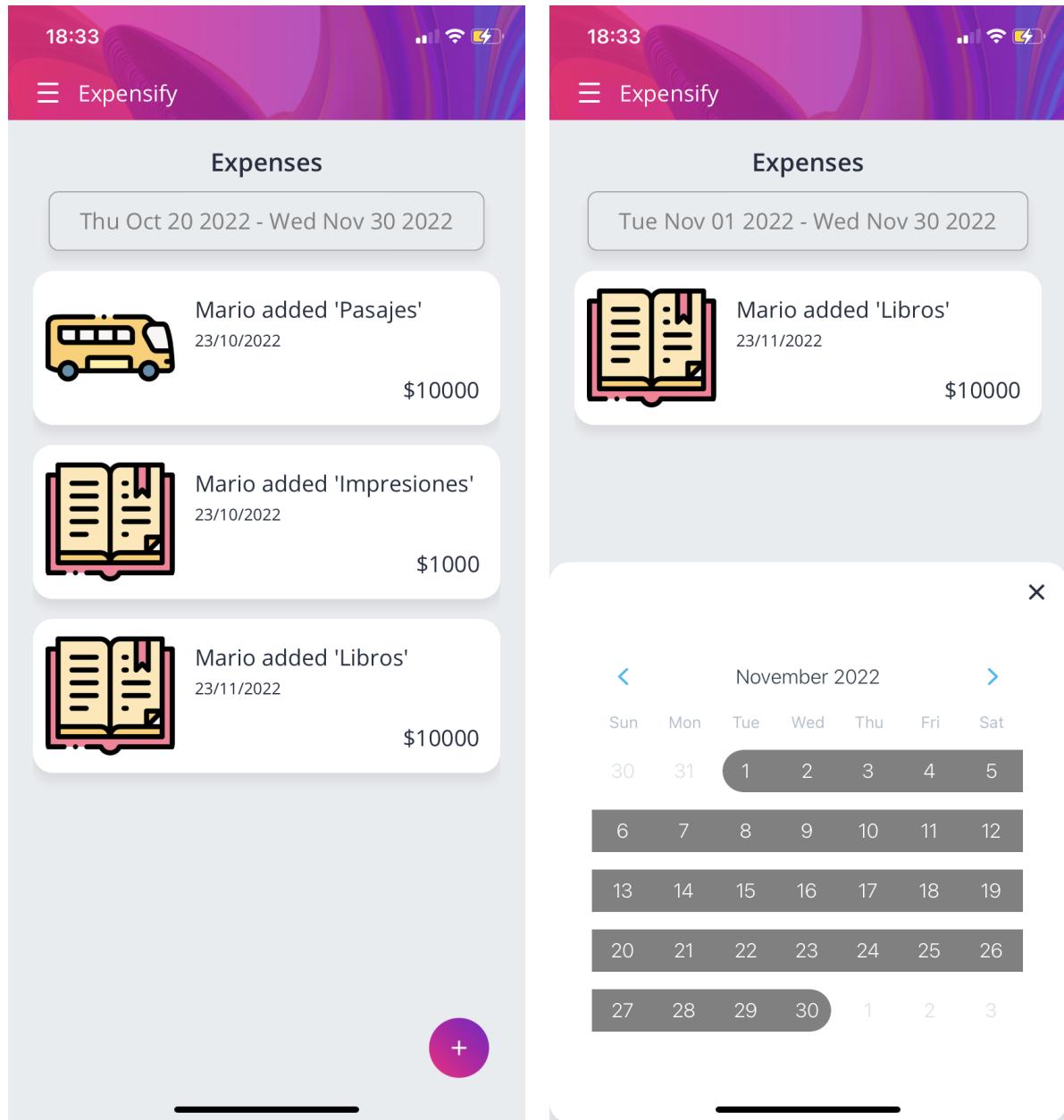


Ilustración 23: Agregar gasto

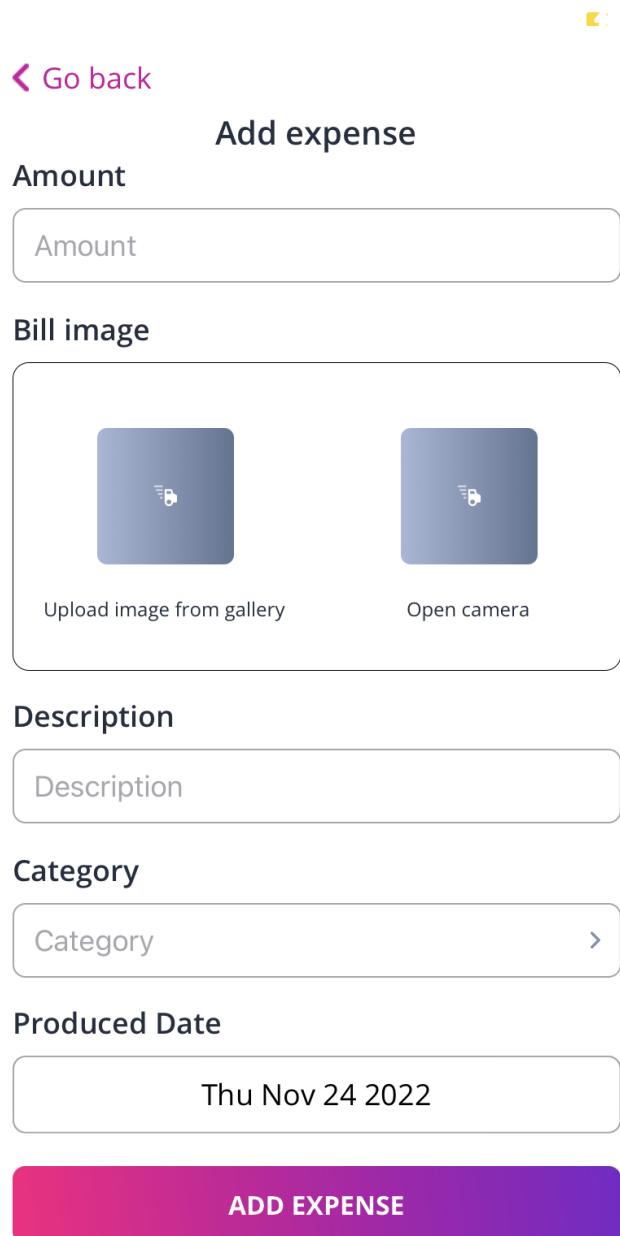


Ilustración 24: Detalles de un gasto

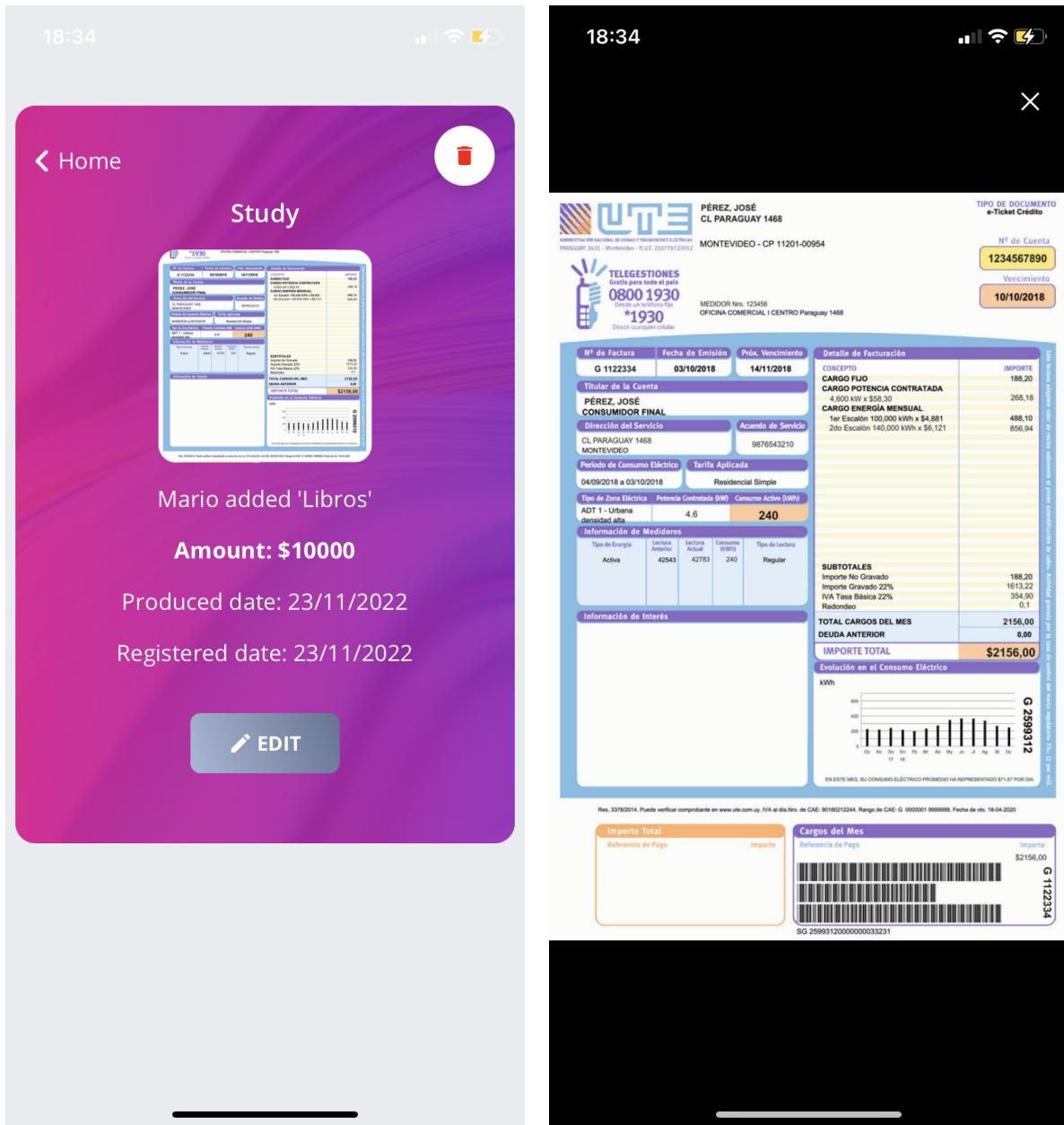


Ilustración 25: Modificar gasto

✖

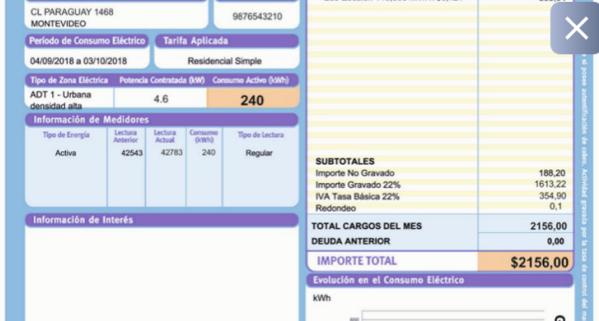
◀ Go back

Edit expense

Amount

10000

Bill image



×

Description

Libros

Category

Study

✖

Produced Date

Tue Nov 22 2022

MODIFY EXPENSE

Ilustración 26: Eliminar gasto

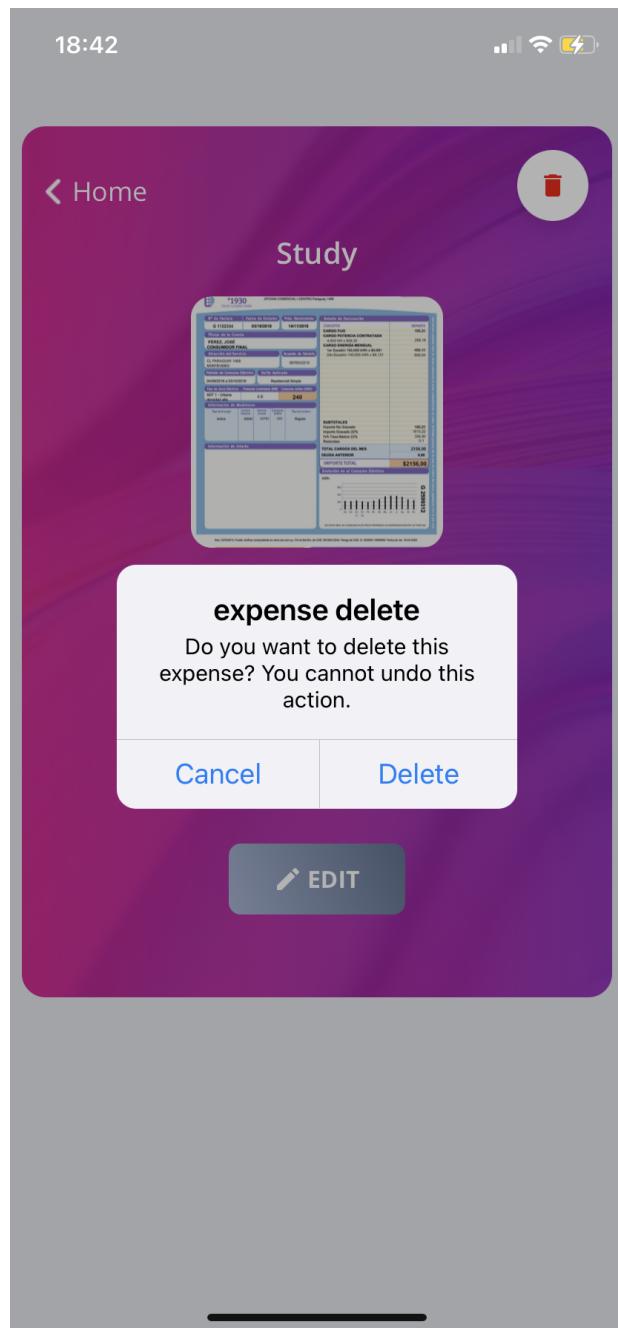


Ilustración 27: Pantalla categorías

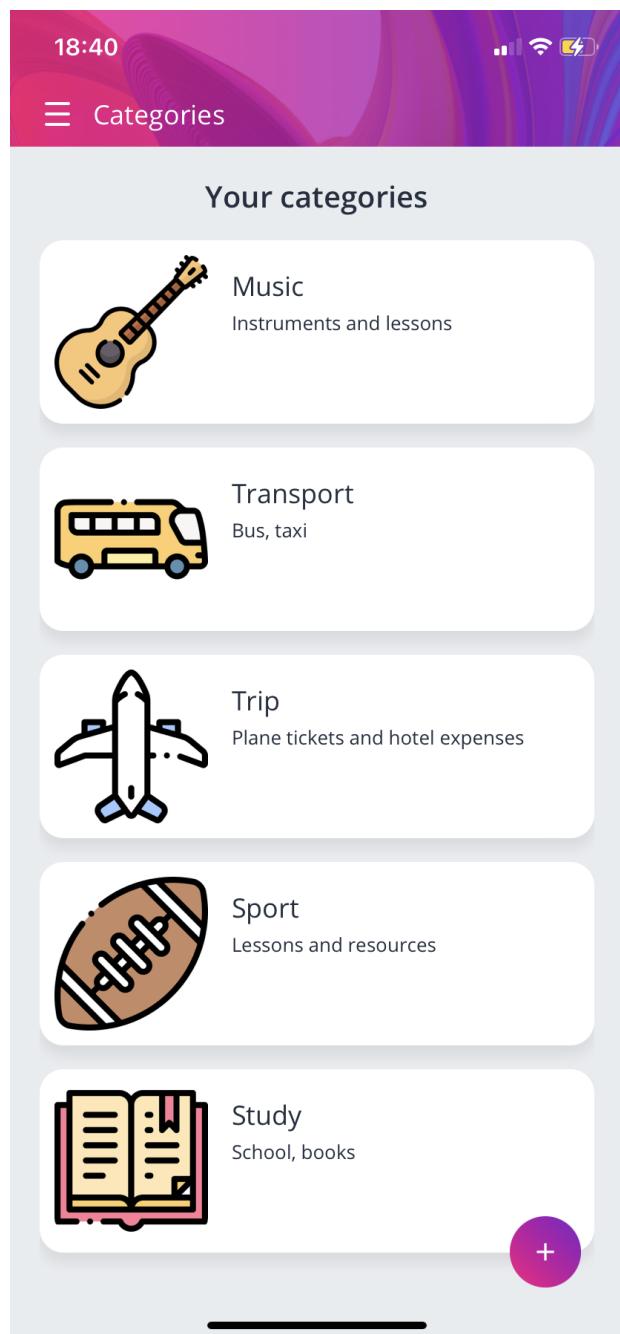


Ilustración 28: Agregar categoría

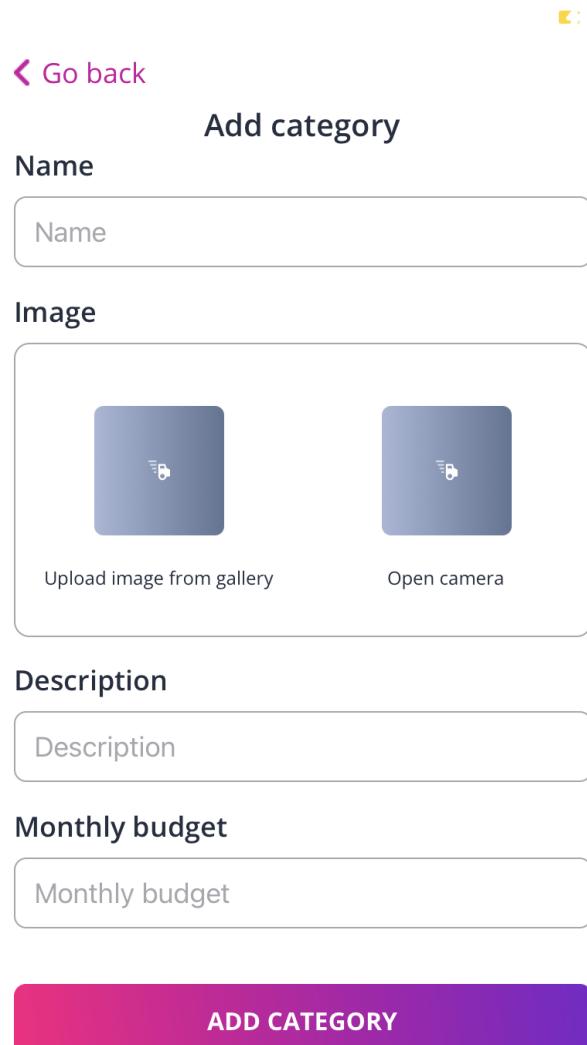


Ilustración 29: Detalles de una categoría

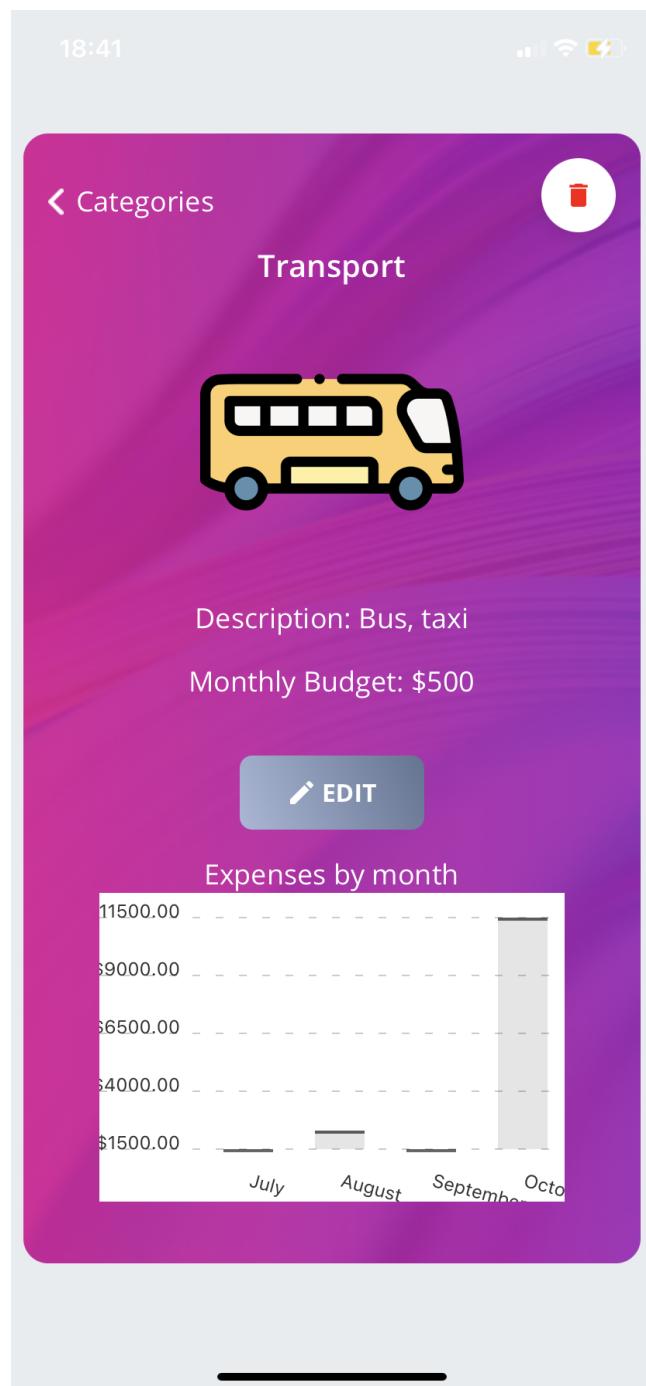


Ilustración 30: Modificar categoría

◀ Go back

Edit category

Name

Image

Description

Monthly budget

MODIFY CATEGORY

Ilustración 31: Eliminar categoría

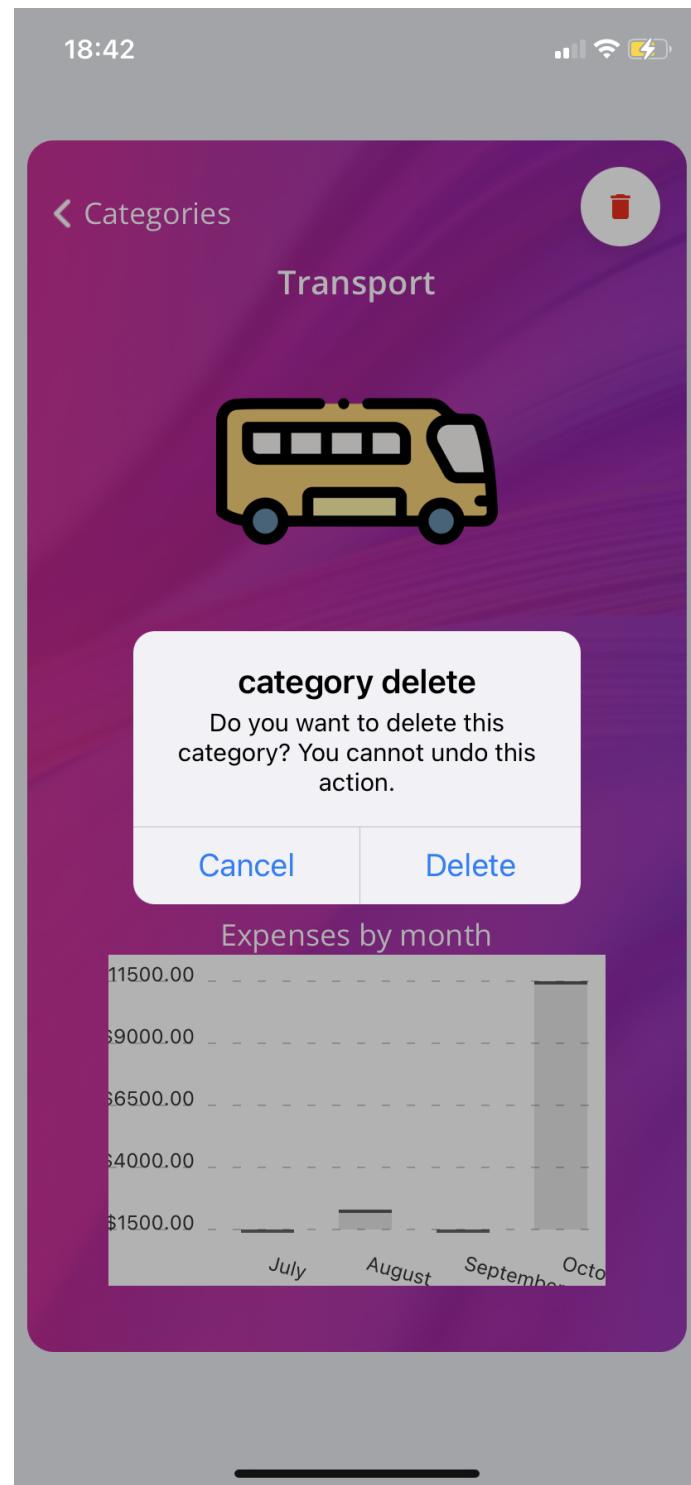


Ilustración 32: Análisis de gastos

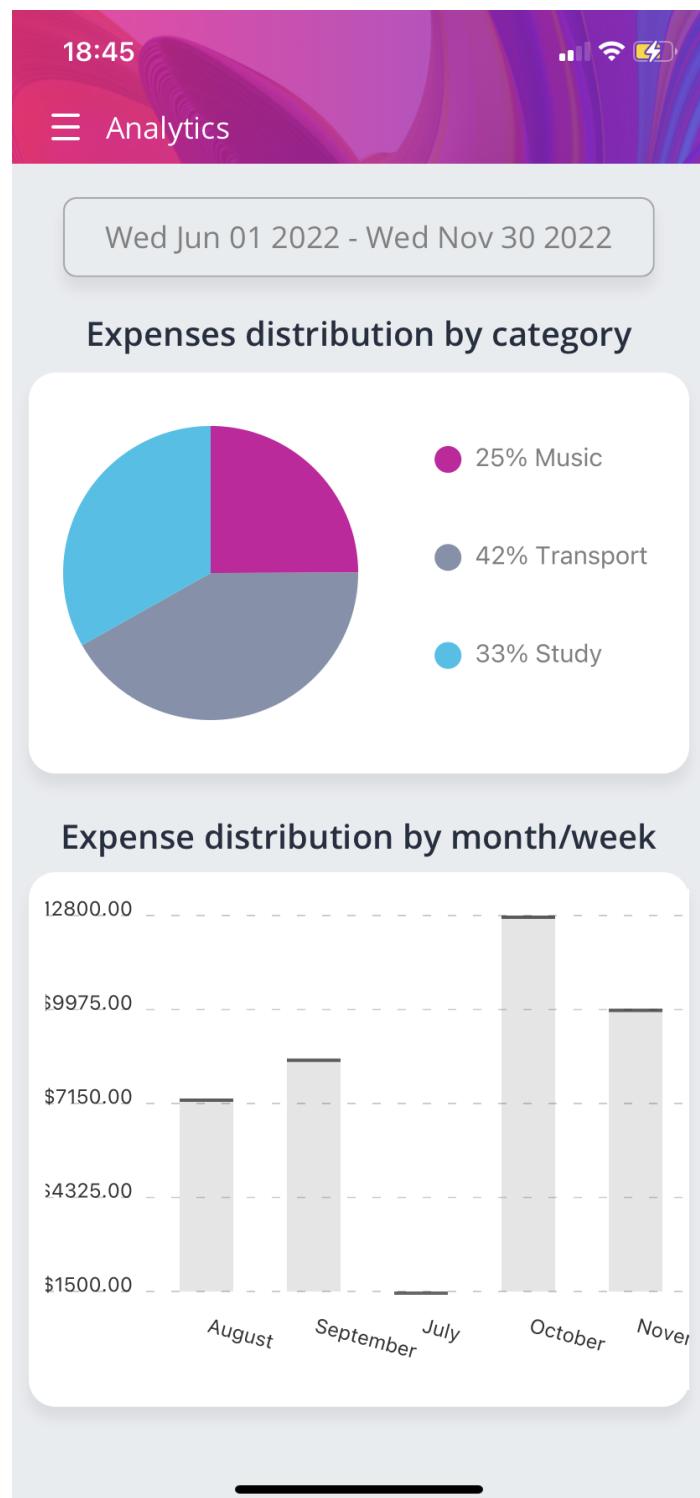


Ilustración 33: Creación de una invitación

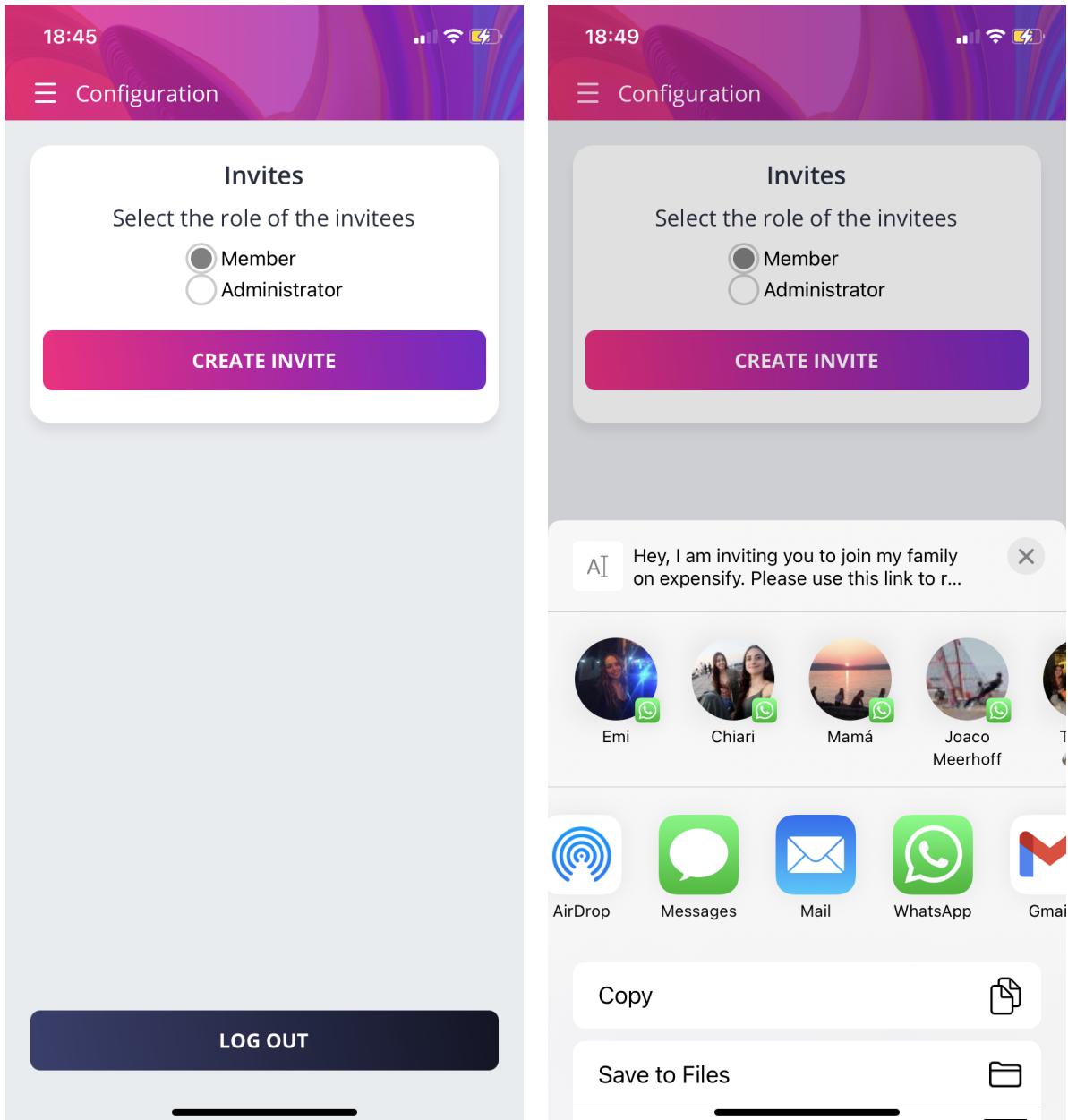


Ilustración 34: Mensaje de error

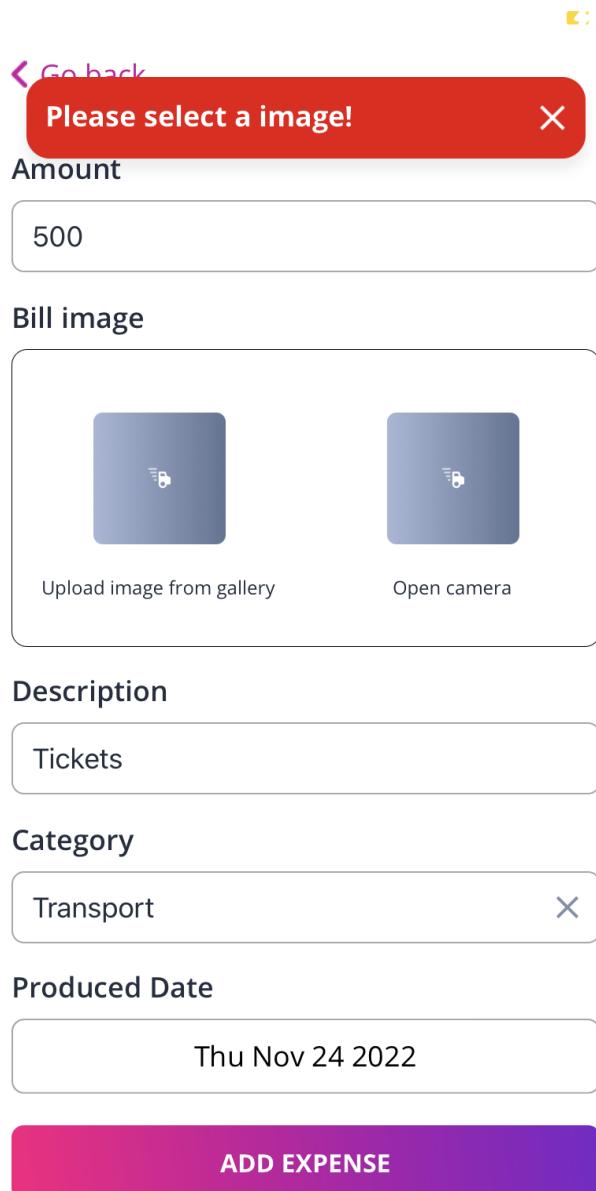


Ilustración 35: Mensaje de éxito

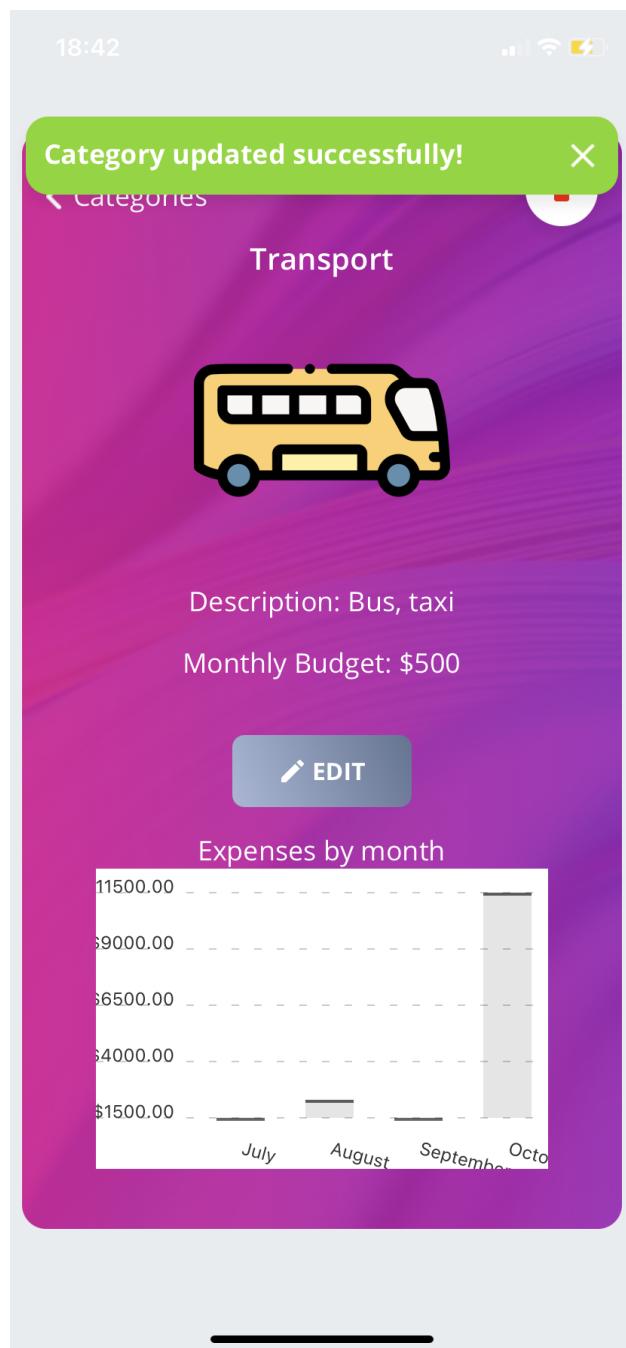
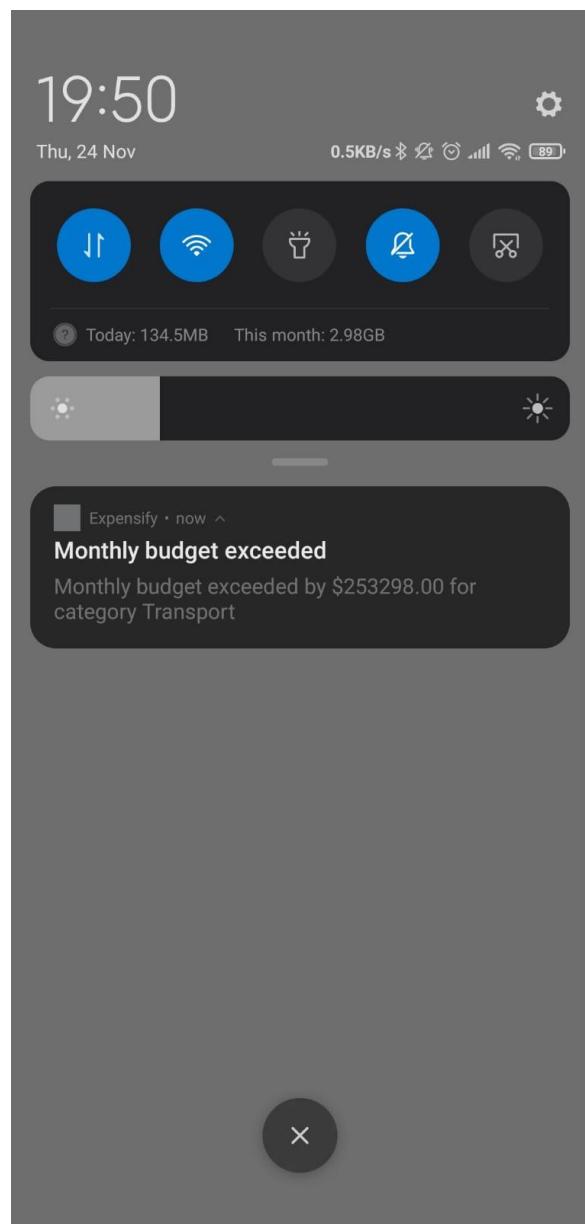


Ilustración 36: Alerta por superación de presupuesto mensual



Anexo 4: Especificación de la API

Con el objetivo de respetar los estándares REST:

- La URI de los endpoints se nombraron en plural
- Se utilizaron guiones para separar palabras compuestas
- Se mantuvo en minúscula las palabras
- Se utilizaron query params en caso de que corresponda

User:

Method	URL	Body	Ok	Error
POST	/users	User	200	400 & 409
Crea un usuario. Devuelve BAD REQUEST si falla la validación y CONFLICT si el usuario ya existe.				
POST	/users/invitations	User	200	400 & 409
Crea un usuario que ingresó a la aplicación por medio de una invitación a cierta familia. Devuelve BAD REQUEST si falla la validación y CONFLICT si el usuario ya existe.				
POST	/users/sign-in	-	200	400
Crea una sesión para un usuario que quiere iniciar sesión. Devuelve BAD REQUEST si falla la validación.				
PUT	/users/token	-	200	404
Actualiza el token de las notificaciones al loguearse el usuario para que le lleguen a su celular. Devuelve NOT FOUND si no encuentra al usuario.				

Family:

Method	URL	Body	Ok	Error
GET	/families/{inviteToken}	-	200	-
Devuelve a la familia que el token de invitación guarda.				
POST	/families/invitations	Invite	200	404
Crea una invitación a la familia. Devuelve BAD REQUEST si falla la validación.				

Expense:

Method	URL	Body	Ok	Error
GET	/expenses	-	200	-
Devuelve todas las expensas de una familia, se pueden filtrar según la fecha.				
GET	/expenses/count	-	200	-
Devuelve la cantidad de expensas.				
GET	/expenses/month	-	200	-
Devuelve las expensas filtradas según el mes.				
POST	/expenses	Expense	200	400 & 404
Crea una expensa. Devuelve BAD REQUEST si falla la validación y 404 si no encuentra la categoría a la que se quiere asociar.				
PUT	/expenses/{id}	-	200	400 & 404
Modifica una expensa. Devuelve BAD REQUEST si falla la validación y 404 si no encuentra la categoría a la que se quiere asociar.				
DELETE	/expenses/{id}	-	200	-
Elimina una expensa.				

Category:

Method	URL	Body	Ok	Error
GET	/categories	-	200	-
Devuelve todas las categorías creadas por una familia.				
GET	/categories/count	-	200	-
Devuelve la cantidad de categorías.				
GET	/categories/{id}/expenses	-	200	-
Devuelve las expensas separadas según la categoría a la que pertenecen.				
POST	/categories	Category	200	400 & 409
Crea una categoría. Devuelve BAD REQUEST si falla la validación y CONFLICT si la categoría ya existe.				
PUT	/categories/{id}	Category	200	400 & 409
Modifica una categoría. Devuelve BAD REQUEST si falla la validación y CONFLICT si la categoría ya existe.				
DELETE	/categories/{id}	-	200	-
Elimina una categoría.				

Anexo 5: Archivo .env

```
AWS_BUCKET_NAME = "backend-category-dev"
AWS_BUCKET_REGION = "us-east-1"
AWS_ACCESS_KEY = "ASIAXAD43ND7KRMFKS7"
AWS_SECRET_ACCESS_KEY = "KWygR5fgPAwdrCYvU0nNBCf48HAMU4ziU1/oVoia"
AWS_SESSION_TOKEN =
"Tw0GZXIvYXdzEL3//////////wEaDJiHdW9j9eaRRz3UKSLBAaDBjXPH1bUVAE1IxKYS7tRWikj62vkD5L+U06+Z
ZCDe/fH5S1mFza+x+rEkFwPZ9KAGaB7txlyvODx7Nj1Np0WV+guRbND5sRJm9BfFrkgVS1/74naIAeNeLueBPr7k6F8
p7I5oKiAxZ7XLe17CBHs2MBgoZLos+spscI3wVS51FyRTchXcXpCn75DXU6KqfyX9CkxP200K4tTAn9t7aNAAaL96p+s
yEnQpDJFOUXhU1AmvWC0MFpmGEHeVj3odMon73NmgYyLa0HX0JWBWtSGkZh3rXrrnUoBMswRsh9YuMVEiH3/rTdzf
+1WfVu+hMDInEGA=="

FRONTEND_ORIGIN = "http://192.168.1.3:3000"
FRONTEND_DOMAIN = "192.168.1.3"

MYSQL_DB_NAME = "db"
MYSQL_DB_USER = "user"
MYSQL_DB_PASSWORD = "wjMZTZ6f6j45GA"
MYSQL_DB_HOST = "localhost"
MYSQL_DB_PORT = "3306"

EMAIL_SERVICE_API_KEY =
"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"
EMAIL_SERVICE_SENDER = "expensify-ort@mail.com"

URL_INVITATION = "expensify://screens/register/"
API_PORT = 3001

SECRET_KEY =
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.GZRMW1CUDVjUTROQm1hNDluOE9rS1FFSnEzTDFNOT12VW9iQ1ZLUH
VYMDlWhhYahJQM0t3bHY4SE1RaHNGdnFjb11ieGxGSHM9IiwiiaXNzIjoicGxheS1zaWxob3VldHR1IiwiZXhwIjoxN
TI2MjgxMDU1LCJpYXQiOjE1MjYyMzc4NTUsImp0aSI6IjhmOGM3YTVmYWRkNTE5MjUxNzQ5NGE4N2Q10DcxZjJjZGUx
ZDkzMdk0TY1MThjNjg2NzExOTc1YjU3M2I5NDB1ZWU2NGY0NDUwYzcxODI3NGZmNzU1MmE2Y2J1NTVmZmZhMWI1ZjA
3ZW10WVkNTE0Y2Y4YTVi0TZ1M2ExYjI0ODRmYT15NjZiYjA0OD1mODIwZjMyMzM5YWVhNjM3NWRkZmU4ZDE4N2E2Nz
BjMzg00DgwZGIyMzQ1ZTFkMzRkYWNjZmY2MTdkMDY1NzU3YmEwZTQzNDg4YWfhZmZmNDNjYWZ1ZGY00TF10DU1YTA0N
WM0NmjjNDY4NGYz0D1mY2YifQ.GwN6TSNd426xpc3Y02eRXHbrmSr_61MMBqrmx660fq5"

MONGO_DB_URI = "mongodb://root:T5wmVuJDe3JF72jfjk45@localhost:27017"
MONGO_DB_NAME = "expensify-logs"
```

```
MONGO_DB_LOGS_COLLECTION = "logs"
```

```
APP_NAME=undefined
```

```
LICENSE_KEY=undefined
```