Escuela Técnica Superior de
Ingeniería Informática

Trabajo de Fin de Grado

Grado en Ingeniería del Software

Departamento de Lenguajes y Sistemas Informáticos

ESTUDIO TECNOLÓGICO Y DESARROLLO DE APLICACIÓN BANCARIA VVBA

Realizado por:

Miguel Ángel Romalde Dorado

Dirigido por:

Juan Antonio Ortega Ramírez

Sevilla, octubre 2024

Tabla de contenidos

Lista de tablas

Lista de ilustraciones

Resumen

Este trabajo pretende analizar y explorar alternativas en el mercado correspondiente al ámbito tecnológico con una aplicación de las mismas en una solución específica, enfocada a la ayuda y facilitación a la población a usar soluciones tecnológicas desde la comodidad de su hogar, se puede dividir el proyecto en dos partes.

La primera se centrará en un estudio de las tecnologías más usadas actualmente en las empresas, se hará tanto un estudio para Front-End como un estudio para Back-End, en el que se verá sus características, a que están enfocadas, ventajas y desventajas, junto con unas tablas comparativas que aportan una mejor visualización, de este estudio se ha seleccionado una tecnología de cada ámbito, estas tecnologías son: ASP .NET Core y Angular, se han elegido estas tecnologías ya que en un principio había una oferta de trabajo la cual requería conocimientos de dichas tecnologías y podían ayudar a la entrada del mundo laboral. Se ha implementado una solución utilizando dichas tecnologías, Banco VVBA.

Por otra parte, para completar el caso de estudio y objetivo del proyecto en su forma práctica, se ha desarrollado una aplicación para la plataforma de escritorio Windows, que, mediante el uso de las tecnologías estudiadas, se ha instaurado los casos de uso core de una aplicación bancaria, estos casos de uso, como puede ser ingresar dinero, retirar dinero, hacer una transferencia, ver las operaciones, etc…, son comunes entre todas las aplicaciones. Este proyecto se ha desarrollado con dos ideas esenciales:

La primera es ser una aplicación de prueba que sirve para que la población que no está acostumbrada a usar aplicaciones bancarias o tienen miedo del uso de la misma, puedan probarla con total libertad y sin ningún riesgo. Por otra parte, la aplicación Banco VVBA se usará para la enseñanza en clases de economía del profesor Juan Antonio Andrés Lalueza.

Abstract

This work aims to analyze and explore alternatives in the market corresponding to the technological field with an application of the same in a specific solution, focused on helping and facilitating the population to use technological solutions from the comfort of your home, the project can be divided into two parts.

The first will focus on a study of the most used technologies currently used in companies, there will be both a study for Front-End and a study for Back-End, in which you will see its characteristics, which are focused, advantages and disadvantages, along with comparative tables that provide a better visualization, this study has selected a technology in each area, these technologies are: ASP .NET Core and Angular, these technologies have been chosen since at first there was a job offer which required knowledge of these technologies and could help the entry of the labor world. A solution has been implemented using these technologies, Banco VVBA.

On the other hand, to complete the case study and objective of the project in its practical form, an application has been developed for the Windows desktop platform, which, through the use of the technologies studied, has established the core use cases of a banking application, these use cases, such as deposit money, withdraw money, make a transfer, view transactions, etc ..., are common to all applications. This project has been developed with two essential ideas:

The first is to be a test application that serves for the population that is not accustomed to using banking applications or are afraid of using it, so that they can try it freely and without any risk. On the other hand, the VVBA Bank application will be used for teaching in economics classes by Professor Juan Antonio Andrés Lalueza.

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor, Juan Antonio Ortega Ramírez, por haber tutorizado este proyecto y por la enorme paciencia y flexibilidad que ha tenido conmigo. Me ha sabido guiar y darme respuesta a las necesidades que surgían a lo largo del desarrollo de este.

También, me gustaría agradecer a todos aquellos que, voluntariamente, participaron en la encuesta inicial del proyecto y en algunas de las pruebas.

1. INTRODUCCIÓN

Este trabajo pretende la realización de un análisis de las tecnologías más usadas por las empresas, en conjunto con aplicación de estas tecnologías analizadas dichas anteriormente.

Por otra parte, para completar el caso de estudio y objetivo del trabajo en su forma práctica, se busca desarrollar una aplicación para la plataforma de escritorio Windows, de forma colateral, con la aplicación se pretende fomentar la independencia de los usuarios a la hora de trabajar con una aplicación de banco y reducir la incertidumbre o el miedo a la utilización de aplicaciones del mismo estilo.

Cabe destacar que se usarán tecnologías actuales, las cuales ayuden al propietario de la aplicación a su entrada a diferentes empresas. Para ello, previamente se ha realizado un estudio detallado sobre la descripción, características, ventajas y desventajas de cada una de las tecnologías existentes de Front-End y Back-End, con el fin de obtener una correcta selección tecnológica específica y adecuada para el desarrollo de la aplicación. En concreto, se ha elegido ASP .NET Core para el Back-End y Angular para el Front-End, dado que son tecnologías actuales que encajan con nuestros estándares y se ha visto una mayor cantidad de ofertas de puestos de trabajo con dichas tecnologías.

Por último, previo a la puesta en marcha de la aplicación bancaria (Banco VVBA), y junto al estudio de las tecnologías anteriormente mencionadas, también se ha realizado una investigación mínima sobre las diferentes aplicaciones bancarias de los cinco grandes bancos existentes a nivel nacional como son Santander, BBVA, CaixaBank, Bankinter y Sabadell y cuyas aplicaciones están puesta a disposición de la sociedad.

ILUSTRACIÓN 1. PRINCIPALES BANCOS EN ESPAÑA

2. ELECCIÓN TENOLÓGICA Y ESTUDIO SOBRE APLICACIONES BANCARIAS

En el siguiente apartado se verá la selección de tecnologías sobre la que vamos a trabajar en el proyecto.

2.1 Selección de tecnologías

Las tecnologías de Front-End y Back-End que se han analizado representan lo último en herramientas y frameworks utilizados por empresas para desarrollar aplicaciones web modernas. La elección de unas sobre otras depende de varios factores como pueden ser: requisitos del proyecto o necesidades de escalabilidad y rendimiento.

2.1.1 Front-End

En este caso se ha elegido utilizar **Angular** en el apartado de Front-End.

ILUSTRACIÓN 2. LOGOTIPO ANGULAR

Se ha seleccionado esta tecnología por las siguientes características:

1. **Desarrollo de SPAs (Single Page Applications)**

Angular es ideal para construir aplicaciones de una sola página (SPA) que proporcionan una experiencia de usuario fluida y rápida sin necesidad de recargar la página.

2. **Estructura Modular**

Utiliza una arquitectura basada en módulos que permite la organización del código, facilitando mantenimiento y escalabilidad del proyecto.

3. **TypeScript**

Angular está construido sobre TypeScript, similar a JavaScript con tipos estáticos y observables. Esto mejora la calidad del código y facilita la detección de errores durante el desarrollo.

4. **Inyección de dependencias**

Angular tiene un sistema de inyección de dependencias que mejora el modularidad y facilita la reutilización de componentes.

5. **CLI (Command Line Interface)**

Angular CLI facilita la creación de nuevos proyectos, la generación de componentes, servicios, y más, además de simplificar diferentes tareas como pruebas o despliegues.

6. **Soporte y comunidad**

Angular es desarrollado y mantenido por Google, lo que garantiza su continuidad y evolución. Además, cuenta con una gran comunidad de desarrolladores y recursos de aprendizaje.

2.1.2 Back-End

Para la parte del Back-End se ha decidido utilizar **ASP.NET Core.**

ILUSTRACIÓN 3. LOGOTIPO ASP .NET CORE

Se ha decidido usar esta tecnología dadas las siguientes características:

1. **Rendimiento**

ASP.NET Core está diseñado para ser un framework de alto rendimiento. Su arquitectura modular y ligera contribuye a mejorar la velocidad y eficiencia de las aplicaciones.

2. **Cross-platform**

A diferencia de su predecesor, ASP.NET Core es completamente multiplataforma, permitiendo que las aplicaciones se ejecuten en Windows, macOS y Linux, aunque dado caso, no es relevante pues se va hacer la aplicación solo para Windows.

3. **Inyección de dependencias integrada**

ASP.NET Core tiene un sistema de inyección de dependencias incorporado que facilita la gestión de dependencias (paquetes Nuggets).

4. **Modularidad y flexibilidad**

   ASP.NET Core permite agregar solo los componentes necesarios con lo que, gracias a esta característica, se puede mejorar mucho el rendimiento y ligereza del proyecto.

5. **Compatibilidad con Microservicios**

   Es ideal para arquitecturas de microservicios, permitiendo la construcción de aplicaciones escalables.

6. **Seguridad**

   ASP.NET Core proporciona características de seguridad robustas, como protección contra ataques CSRF.

7. **Ecosistema .NET**

   Se integra perfectamente con el ecosistema .NET, permitiendo el uso de bibliotecas y herramientas existentes y aprovechando las capacidades de C#.

Además de todas las características mencionadas anteriormente, otro motivo importante es que, para hacer este TFG, era necesario usar tecnologías que ayudaran personalmente a la incorporación al mundo laboral. Son tecnologías nuevas y de las más usadas, ambas tienen un gran rendimiento y un soporte a largo plazo, por lo que son ideales para la realización de este proyecto.

2.2 Estudio sobre las aplicaciones bancarias

2.2.1 Banco La Caixa

- **Descripción:** La página web de “La Caixa” es la plataforma digital que ofrece a sus usuarios una amplia gama de servicios bancarios en línea, incluyendo gestión de cuentas, transferencias, pagos, inversiones y servicios de financiación, tanto para particulares como para empresas. Además, proporciona información detallada sobre los productos y servicios de CaixaBank, así como acceso a recursos de apoyo y contacto.

ILUSTRACIÓN 4. LOGOTIPO CAIXA

- **Características**
  - Gestión de cuentas: permite a los usuarios gestionar sus cuentas bancarias, realizar transferencias y ver movimientos.
  - Servicios de inversión: ofrece información y herramientas para la gestión de inversiones y fondos.
  - Préstamos y financiación: proporciona opciones de financiación y préstamos con detalles sobre condiciones y solicitudes.
  - Pagos y transferencias: facilita el pago de facturas y la realización de transferencias tanto nacionales como internacionales.
  - Soporte y atención al cliente: incluye recursos de ayuda, FAQs, y contacto con atención al cliente.
  - Seguridad: implementa medidas de seguridad avanzadas para proteger las transacciones y datos de los usuarios.
- **Ventajas**
  - Comodidad: permite realizar operaciones becarias desde cualquier lugar y en cualquier momento.
  - Ahorro de tiempo: los usuarios pueden gestionar sus finanzas sin necesidad de acudir a una sucursal física.
  - Acceso a información: ofrece un acceso fácil y rápido a información sobre productos financieros y servicios.
  - Seguridad: alta seguridad en las transacciones, protegiendo los datos personales y financieros de los usuarios.
- **Desventajas**
  - Dependencia tecnológica: requiere una conexión a internet y dispositivos adecuados para el acceso.
  - Curva de aprendizaje: algunos usuarios pueden encontrar complicado navegar y utilizar todas las funciones disponibles.
  - Limitaciones en la interacción personal: la falta de interacción cara a cara puede ser un inconveniente para aquellos que prefieren el trato personal en sus gestiones bancarias.

2.2.2 Banco Santander

- **Descripción**: la página web de Santander proporciona una plataforma completa para la gestión de servicios bancarios y financieros. Está diseñada para facilitar el acceso a cuentas personales y empresariales, realizar transacciones y gestionar productos financieros de manera eficiente.

ILUSTRACIÓN 5. LOGOTIPO SANTANDER

- **Características:**
  - Acceso a cuentas y productos: permite a los usuarios consultar saldos, movimientos y gestionar tarjetas, préstamos y seguros.
  - Transacciones en línea: facilita la realización de transferencias, pagos de recibos e impuestos, y envío de dinero a través de Bizum.
  - Seguridad: implementa medidas de seguridad avanzadas como la autenticación con huella dactilar y la generación de códigos para operaciones seguras.
  - Herramientas financieras: ofrece simuladores y opciones de financiación como fraccionamiento de compras y pagos a plazos.
- **Ventajas:**
  - Comodidad y accesibilidad: los usuarios pueden gestionar sus finanzas desde cualquier lugar y en cualquier momento.
  - Amplia gama de servicios: proporciona una gran variedad de servicios financieros y herramientas de gestión.
  - Integración móvil: la app de Santander permite realizar todas las operaciones bancarias desde dispositivos móviles, incluyendo pagos y retiradas de efectivo.
  - Alertas personalizadas: los usuarios pueden configurar alertas para mantenerse informados sobre movimientos y transacciones.
- **Desventajas:**
  - Curva de aprendizaje: la complejidad de algunas funciones puede requerir tiempo para que los usuarios menos familiarizados se adapten.

- Pesadez del sitio: algunas funcionalidades pueden resultar lentas o complicadas debido a la amplia cantidad de servicios y herramientas disponibles.
- Menor flexibilidad: la página web puede resultar menos flexible y más estructurada.

2.2.3 Banco BBVA

- **Descripción:** la página web de BBVA ofrece una plataforma digital integral que permite a los usuarios gestionar sus finanzas personales y empresariales en línea. Proporciona acceso a una variedad de servicios bancarios, incluidos cuentas, préstamos, inversiones y herramientas de planificación financiera.

ILUSTRACIÓN 6. LOGOTIPO BBVA

- **Características:**
  - Banca online y móvil: acceso a servicios bancarios a través de la web y aplicaciones móviles.
  - Seguridad: múltiples capas de seguridad, incluyendo autenticación de dos factores y cifrado SSL.
  - Gestión de cuentas: visualización y gestión de cuentas, transferencias y pagos.
  - Inversiones: acceso a productos de inversión y herramientas de análisis.
  - Atención al cliente: soporte a través de chat, teléfono y correo electrónico.
- **Ventajas**
  - Comodidad: permite a los usuarios a realizar transacciones y gestionar sus finanzas desde cualquier lugar.
  - Seguridad: protocolos de seguridad robustos para proteger la información del usuario.
  - Variedad de servicios: amplia gama de servicios financieros disponibles en línea.
  - Interfaz intuitiva: Diseño fácil de usar.

- **Desventajas:**
  - Curva de aprendizaje: algunos usuarios pueden encontrar complejas las funcionalidades avanzadas.
  - Requiere conexión a internet: la dependencia de internet puede ser un inconveniente en áreas con mala conectividad.
  - Limitaciones regionales: algunos servicios pueden no estar disponibles en todas las regiones.

2.2.4 Banco Bankinter

- **Descripción:** La página web de Bankinter es una plataforma digital que ofrece servicios bancarios a individuos, empresas y clientes de banca privada. Está diseñada para proporcionar acceso a una amplia gama de productos y servicios financieros, desde cuentas bancarias y tarjetas hasta inversiones y seguros.

ILUSTRACIÓN 7. LOGOTIPO BANKINTER

- **Características:**
  - Interfaz amigable: navegación sencilla y accesible
  - Acceso seguro: inicia sesión en el área privada con opciones de seguridad mejoradas.
  - Servicios integrados: herramientas para la gestión de cuentas, transferencias y pagos.
  - Calculadoras financieras: diversas herramientas para calcular hipotecas, inversiones y seguros.
  - Información detallada: acceso a información y comparación de productos financieros.
  - Soporte al cliente: opciones de contacto y asistencia en línea.
- **Ventajas:**
  - Amplia gama de herramientas: calculadora y simuladores que facilitan la planificación financiera.
  - Acceso móvil: versión optimizada para dispositivos móviles.

- Personalización: ofertas y productos adaptados a las necesidades del usuario.
- Seguridad: protocolos robustos para la protección de datos y transacciones.
- Facilidad de uso: interfaz intuitiva que facilita la navegación y el acceso a servicios.

Desventajas:

- Curva de aprendizaje: puede ser abrumador para nuevos usuarios debido al amplia gama de servicios.
- Dependencia de internet: requiere una conexión estable para el uso óptimo de todas sus funcionalidades.
- Actualizaciones necesarias: requiere actualizaciones periódicas del navegador para garantizar la seguridad y funcionalidad.

2.2.5 Banco Sabadell

Descripción: la página web de Banco Sabadell es una plataforma digital que ofrece una amplia gama de servicios bancarios y financieros tanto para clientes particulares como para empresas. A través de su portal, los usuarios pueden acceder a cuentas bancarias, tarjetas, hipotecas, préstamos, inversiones, seguros, etc.

ILUSTRACIÓN 8. LOGOTIPO SABADELL

Características:

- Cuentas y tarjetas: los usuarios pueden abrir y gestionar cuentas, así como solicitar tarjetas de débito y crédito.
- Préstamos e hipotecas: simulación y solicitud de préstamos personales e hipotecas.
- Servicios digitales: incluye servicios como Bizum para transferencias instantáneas y la posibilidad de gestionar cuentas y pagos a través de la aplicación móvil.
- Gestión de inversiones y ahorros: opciones para gestionar inversiones y planes de ahorro.

- Atención al cliente: asistencia a través de chat en vivo, teléfono y formularios de contacto.

Ventajas:

- Acceso rápido y fácil: los usuarios pueden abrir cuentas y gestionar sus finanzas en menos de 10 minutos a través de su móvil.
- Variedad de servicios: ofrece una amplia gama de productos financieros adaptados a diferentes necesidades.
- seguridad: dispone de medidas de seguridad robustas para proteger las transacciones y la información del usuario.
- Integración con Bizum: facilita transferencias rápidas y sencillas.
- Alertas y notificaciones: los usuarios reciben notificaciones personalizadas sobre sus transacciones y movimientos bancarios.

Desventajas:

- Curva de aprendizaje: algunos usuarios pueden encontrar compleja la navegación y utilización inicial del sitio web.
- Limitaciones en servicios profesionales: algunas cuentas, como las de autónomos, tienen restricciones específicas.
- Menor personalización: aunque ofrece muchas herramientas, puede no ser tan flexible o personalizada como el contacto directo en una sucursal.

2.2.6 Tabla comparativa de las aplicaciones bancarias.

SERVICIOS	CaixaBank	Santander	BBVA	Bankinter	Sabadell
Características					
Gestión de cuentas	✓	✓	✓	✓	✓
Préstamos y financiación	✓	✓	✓	✓	✓
Pagos y transferencias	✓	✓	✓	✓	✓
Soporte de atención al cliente	✓	✓	✓	✓	✓
Seguridad	✓	✓	✓	✓	✓
Información detallada	✓	✓	✓	✓	✓
Ventajas					
Comodidad	✓	✓	✓	✓	✓
Ahorro de tiempo	✓	✓	✓	✓	✓
Integración móvil	✓	✓	✓	✓	✓
Personalización	✗	✓	✓	✓	✓
Desventajas					
Dependencia tecnológica	✓	✓	✓	✓	✓
Pesadez del sitio	✗	✓	✗	✗	✗
Limitaciones regionales	✗	✗	✓	✗	✗
Limitaciones profesionales	✗	✗	✗	✗	✓

TABLA 1. TABLA COMPARATIVA BANCOS

3. ALCANCE DEL PROYECTO

Antes de comenzar con la parte práctica del trabajo, es conveniente hacer un estudio del propio desarrollo del producto, para lograrlo se ha hecho uso de tres prácticas estándar extendidas en la industria del desarrollo de software, la elicitación de requisitos, la evaluación de riesgos y un pequeño análisis de viabilidad, que sirve para proporcionar un poco más de información sobre qué usuarios podrían llegar a hacer uso de la aplicación.

3.1 Elicitación de requisitos del producto

3.1.1 Requisitos de información

ID	Descripción	Prioridad	Criterio de aceptación
RI-01	Debe almacenar los siguientes datos: nombre, apellidos, nombre de usuario, contraseña, avatar, dirección de correo electrónico e IBAN.	Crítica	La base de datos debe permitir almacenar los datos mencionados y consultarlos.
RI-02	Debe permitir almacenar, descripción, cantidad, persona que hace, persona que recibe y fecha sobre una transferencia.	Critica	Todas las transferencias deben tener estos datos.
RI-03	Debe almacenar un registro de todas las transferencias del usuario.	Alta	Debe poder revisar en todo momento sus transferencias, así como sus movimientos.
RI-04	Debe almacenar datos de ingresos y extracciones, de dinero a la cuenta del usuario, con su fecha y cantidad correspondiente.	Alta	Debe poder revisar en todo momento los ingresos y extracciones de la cuenta del usuario.

TABLA 2. REQUISITOS DE INFORMACIÓN

3.1.2 Requisitos funcionales

A continuación, se verán los requisitos funcionales de nuestra aplicación.

ID	Descripción	Prioridad	Criterio de aceptación
RF-01	Debe permitir un inicio de sesión para el acceso a los datos del usuario.	Crítica	Debe controlarse el acceso a los datos mediante la sesión activa
RF-02	Debe permitir la modificación o eliminación de los datos de la cuenta de un usuario.	Critica	El usuario debe tener control sobre los datos de su cuenta.
RF-03	Debe permitir al usuario hacer ingresos o extracciones en la cuenta.	Alta	Debe existir un formulario para llevar a cabo estas acciones.
RF-04	Debe posibilitar hacer transferencias de una cuenta a otra registrada en la aplicación.	Alta	Las transferencias tendrán un máximo y un mínimo de dinero, nunca podrá ser superior al dinero íntegro en la cuenta.
RF-05	Debe tener una lista de movimientos para la revisión de estos por parte del usuario.	Alta	Se hará una lista paginada la cual sea agradable y sencilla.
RF-06	Debe poder revisar en detalle los datos de cualquier movimiento en la cuenta.	Alta	El usuario puede ver en cualquier momento los datos de los movimientos de su cuenta.
RF-07	Debe mostrar información de contacto del administrador de la aplicación.	Baja	Ha de existir un apartado que cuente con la dirección de correo electrónico del administrador.
RF-08	El administrador debe poder modificar y eliminar cualquier dato de la aplicación.	Media	El administrador tiene total acceso y poder sobre la aplicación.
RF-09	Debe poder exportar los datos de los movimientos de una cuenta en formato PDF.	Media	El usuario puede exportar los datos de los movimientos de la cuenta.

TABLA 3. REQUISITOS FUNCIONALES

3.1.3 Requisitos no funcionales

3.1.3.1 Requisitos de calidad

En este apartado se observa los requisitos de calidad.

ID	Descripción	Prioridad	Criterio de aceptación
RC-01	La disponibilidad del sistema será la más alta posible.	Alta	El sistema deberá ser estable, sin problemas y cierres inesperados.
RC-02	El sistema deberá ser rápido y eficiente para una mayor facilidad del uso.	Media	El sistema será rápido para un mejor uso de este.

TABLA 4. REQUISITOS DE CALIDAD

3.1.3.2 Requisitos de implementación

En este apartado se expondrán los requisitos de implementación.

ID	Descripción	Prioridad	Criterio de aceptación
RIM-01	Se implementará en una versión de escritorio para escritorio (Windows).	Alta	Será desarrollada en Angular y ASP .NET Core.
RIM-02	Empleará una base de datos para almacenar los datos de la aplicación.	Media	Será una base de datos relacional y contendrá todos los datos de la aplicación.

TABLA 5. REQUISITOS DE IMPLEMENTACIÓN

3.1.3.3 Requisitos de seguridad

En este apartado se verán los requisitos de seguridad necesarios en la aplicación.

ID	Descripción	Prioridad	Criterio de aceptación
RS-01	La aplicación no tendrá ningún código ejecutable salvando librerías de terceros.	Media	Aunque es código libre, el código oficial de la aplicación ha de ser protegido.
RS-02	No se podrá modificar movimientos por parte del usuario, solo por parte del administrador y en caso de ser necesario.	Alta	La aplicación mantendrá la integridad de los datos del usuario.
RS-03	Debe garantizar la confidencialidad, integridad y autenticidad de la información transmitida.	Crítica	Especialmente los datos del usuario estarán protegidos de forma robusta y no saldrán de la aplicación.
RS-04	No se debe ejecutar operaciones con privilegios en modo súper usuario.	Alta	No se debe solicitar el modo administrador (root) en ningún momento.
RS-06	Se validarán los formularios para evitar inyecciones de código malicioso.	Critica	Todos los formularios contarán con validación.
RS-07	La aplicación no contendrá ningún software malicioso o perjudicial para el dispositivo del usuario.	Alta	La aplicación no contendrá ningún software malicioso, con fines lucrativos o dañinos.

TABLA 6. REQUISITOS DE SEGURIDAD

3.2 Riesgos

La clasificación de los riesgos que pueden afectar a la aplicación se estimará multiplicando la probabilidad por el impacto del riesgo. Para determinar el impacto de un riesgo se tienen en cuenta tres parámetros, la modificación de alcance de la aplicación,

repercusión en la calidad y necesidad de revisiones de la planificación. Para realizar la estimación se ha utilizado la siguiente figura:

Matriz de Probabilidad e Impacto		Impacto				
		Muy bajo 0.05	Bajo 0.1	Moderado 0.2	Alto 0.4	Muy Alto 0.8
Probabilidad	Muy alta 0.9	0.045	0.09	0.18	0.36	0.72
	Alta 0.7	0.035	0.07	0.14	0.28	0.56
	Moderada 0.5	0.025	0.05	0.1	0.2	0.4
	Baja 0.3	0.015	0.03	0.06	0.12	0.24
	Muy Baja 0.1	0.005	0.01	0.02	0.04	0.08

Riesgo Bajo ---- Riesgo Moderado ---- Riesgo Alto

ILUSTRACIÓN 9. MATRIZ DE PROBABILIDAD E IMPACTO

Donde la probabilidad e impacto máxima es 0.72.

Para su evaluación, primero se consideran los riesgos entre 10% y 90% de probabilidades de que ocurra, luego se decidirá el impacto de ese riesgo, y se hará una multiplicación entre la columna y la fila correspondiente para obtener la puntuación.

ID	Descripción	Probabilidad	Impacto	Puntuación	Riesgo
1	Estimación incorrecta de la planificación.	Moderada	Muy Alto	0.4	Alto
2	Subestimar el alcance.	Alta	Alto	0.28	Alto
3	Cambio de requisitos al avance del proyecto.	Media	Alto	0.2	Medio
4	Incompatibilidad en funcionalidades tecnológicas.	Baja	Bajo	0.03	Bajo
5	Alta complejidad en el desarrollo del modelo.	Medio	Alto	0.2	Medio
6	Bajo rendimiento en la aplicación.	Bajo	Medio	0.06	Medio
7	Baja aceptación de la aplicación.	Bajo	Bajo	0.03	Bajo

TABLA 7. TABLA DE RIESGOS

Como se puede observar en la tabla 5, hay varios riesgos a considerar en el desarrollo de la aplicación, aunque se ve que la mayoría de ellos suponen un riesgo bajo/medio, hay algunos riesgos altos, aunque sean en la parte organizativa de la aplicación.

3.3 Viabilidad

En este caso se ha hecho una encuesta anónima en la cual hay 7 preguntas realizadas mediante la plataforma de Google Forms. Se debe destacar que en dicha en cuentas han participado 75 personas, de las cuales se han revelado los siguientes resultados.

Es importante mencionar que el profesor Juan Antonio Andrés Lalueza, perteneciente al Centro Educativo Altair, ubicado en la provincia de Sevilla, utilizará este proyecto, es decir, esta aplicación bancaria VVBA para sus clases de economía que imparte en dicho centro.

A continuación, con respecto a la gráfica 1, se puede observar que un 41´9%, es decir, 31 de 75 persona se muestran “satisfechos” con el uso de las aplicaciones bancarias de manera general. Por otro lado, hay un 2´7% y un 16,2% que se muestran “muy insatisfechos” e “insatisfechos” por lo que existe un número elevado de usuarios a los cuales, les resulta complicado el manejo de las aplicaciones bancarias.

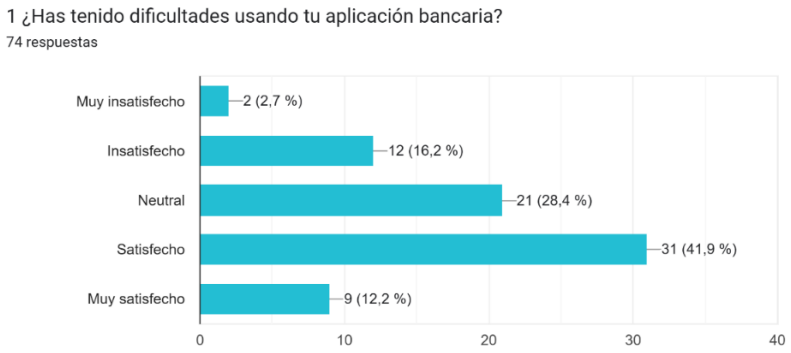

ILUSTRACIÓN 10. ENCUESTA PREGUNTA 1

En relación a la gráfica 2, existe un 60,8% que se muestra “satisfecho” con las aplicaciones bancarias en general, por lo que se puede intuir que el uso de estas aplicaciones, facilita a los usuarios las operaciones y trámites bancarios.

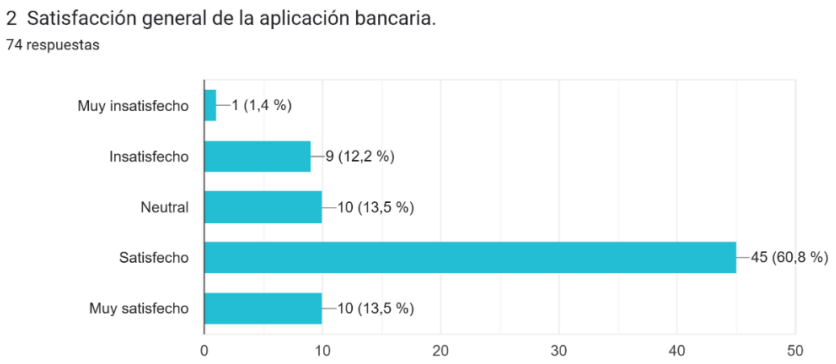

ILUSTRACIÓN 11. ENCUESTA PREGUNTA 2

En la gráfica 3, los usuarios se muestran “satisfechos” con un 52´7% y “muy satisfecho” con un 27%, esto significa que un gran número de personas sienten seguridad a la hora de realizar transacciones a través de las aplicaciones bancarias. El resto de personas se muestran “neutral” o se sienten inseguras a la hora de realizar estas operaciones.

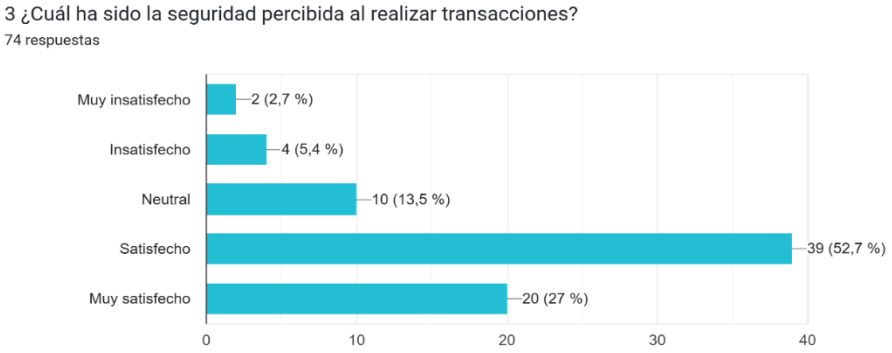

ILUSTRACIÓN 12. ENCUESTA PREGUNTA 3

En la gráfica 4, se puede observar que un 81´1% de las personas no necesitan una aplicación bancaria de prueba para poder entender mejor su uso, por otro lado, un 20´3% necesitarían una aplicación de prueba que les simule cada una de las operaciones que pueden realizar en una aplicación bancaria de manera real, para que de esta manera puedan practicar.

Se debe destacar que la muestra obtenida ha sido pequeña, por lo que estos datos podrían fluctuar dependiendo de número de personas, edad o la población a la que se ha dirigido la encuesta.

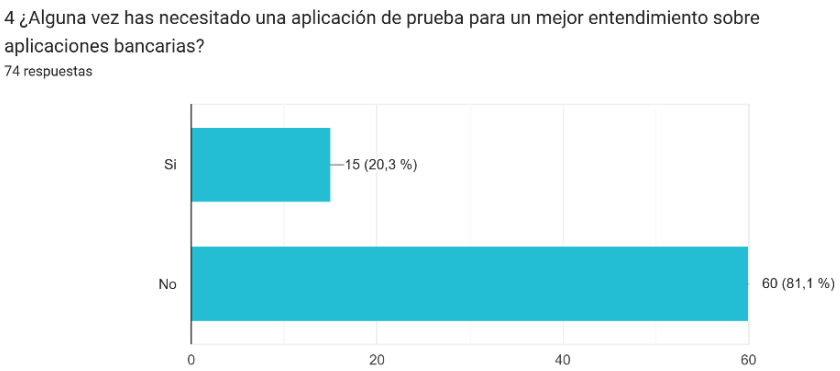

ILUSTRACIÓN 13. ENCUESTA PREGUNTA 4

En la gráfica 5, se puede observar que como un 39´7% de los usuarios que han realizado la encuesta se muestran “neutral”, esto significa que durante el uso de las aplicaciones bancarias todavía no se han encontrado con alguna incidencia o problemas técnicos que los lleve a valorar este aspecto. Aun así, se puede observar que un 24´7% se muestran “insatisfechos”, son usuarios que, ante un incidente o problemas técnicos, el banco a través de la aplicación bancaria no les ha podido ofrecer una solución. Por otro lado, un 28´8% se muestra “satisfecha” con las soluciones obtenidas a través de la aplicación.

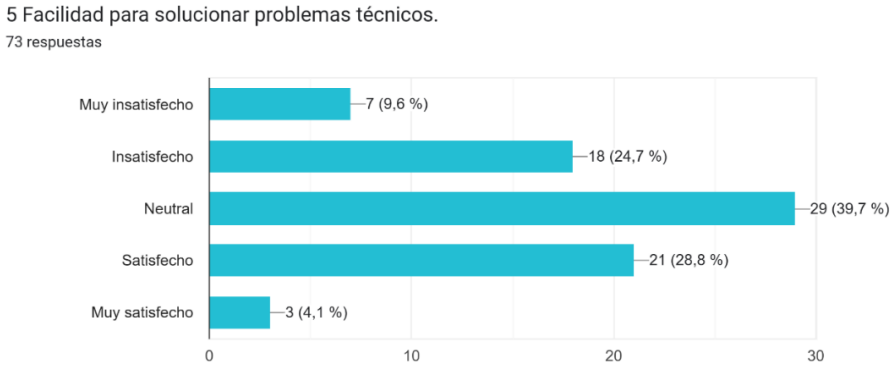

ILUSTRACIÓN 14. ENCUESTA PREGUNTA 5

En la gráfica 6, se puede percibir que un 62´2% de los usuarios que han realizado la encuesta, no necesitan tener una aplicación bancaria, sin embargo, hay un 41´9% que si desearía obtener una aplicación bancaria de prueba para practicar o simular sus operaciones bancarias.

Cabe destacar que, al presentar una muestra pequeña de participantes, estos valores pueden variar dependiendo de la cantidad de participantes que interactúen y según a la población que vaya dirigida la encuesta. Aún así se pretende mostrar de manera general unos resultados previos para realizar un análisis.

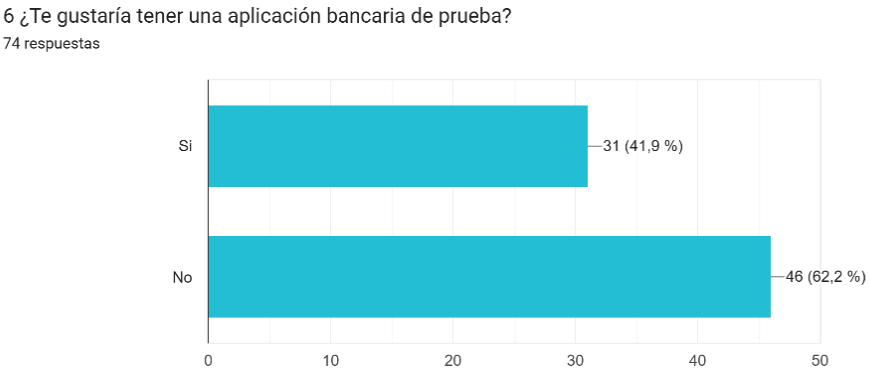

ILUSTRACIÓN 15. ENCUESTA PREGUNTA 6

Por último, en la gráfica 7, existe un alto porcentaje de usuarios 83´8% que no invertiría 2€ para probar una aplicación de prueba, por otro lado, un 17´6% si pagaría este dinero por poseer una aplicación bancaria de prueba que les permitiera aprender cada uno de los movimientos u operaciones bancarias.

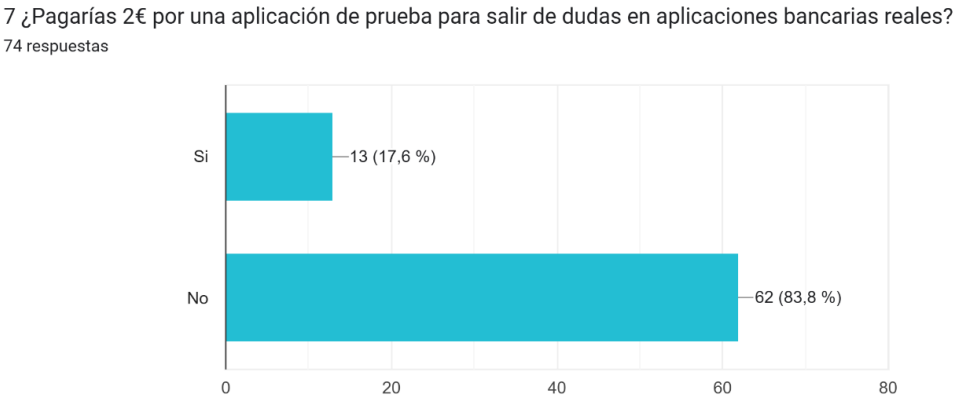

ILUSTRACIÓN 16. ENCUESTA PREGUNTA 7

3.4 Objetivos de la aplicación

El objetivo principal del trabajo es la investigación de las tecnologías más útiles ya que tienen un mayor porcentaje de uso en las empresas y una aplicación de estas tecnologías basada en una solución especifica. Esta solución contará del desarrollo de una aplicación web orientada a ser un ejemplo general de un banco el cual tiene todas las funcionalidades más usadas, para que los usuarios, puedan practicar la utilización de aplicaciones del mismo estilo y haya una mayor comprensión general de estas.

4. VERSIÓN DE COSTES

4.1 Coste salarial del desarrollo

En este apartado se desglosan los costes salariales del equipo de desarrollo que contempla desde la primera semana de vida del proyecto hasta la presentación del proyecto, en este caso, 15 semanas. Ha habido una cantidad de 155,5 horas de trabajo en el proyecto.

	Sueldo por hora	
Rol	Bruto	Neto
Project Manager	23,50€	16,45€

TABLA 8. COSTES DESARROLLO DE EQUIPO

El coste de desarrollo es el número de horas que se ha trabajado por el sueldo por hora del desarrollador: 155,5 * 23,50 = 3.654,25€

A este sueldo, se le ha sumado el coste de la Seguridad Social del trabajador que es el 28%, por lo que el total del coste salarial es: 3.654,25 + 1.023,19 = **4.677,44€**

4.2 Costes derivados del desarrollo

Como su propio nombre indica, aquí se describirán y definirán los costes derivados del desarrollo, en el mismo periodo de tiempo que se han definido hasta la puesta en producción, 15 semanas.

Tipo de gastos	Gasto semanal	Gasto mensual
Gasto eléctrico	15€	60€
Gasto de internet	15€	60€

TABLA 9. COSTES DERIVADOS EN EL DESARROLLO

Estos gastos se deben a la compensación de que sea un trabajo remoto.

Evento	Duración (semanas)	Coste total
Preparación del proyecto	4	60€
Sprint 1	3	45€
Sprint 2	3	45€
Sprint 3	3	45€
Preparación	2	30€
Total	**15**	**225€**

TABLA 10. COSTES DERIVADOS EN EL DESARROLLO TOTALES

4.3 Costes de licencias

A continuación, se tratará el desglose de los costes asociados a las licencias usadas durante el periodo de desarrollo. Teniendo en cuenta los precios sacados de la página oficial de Microsoft, y obteniendo el Microsoft 365 E3 EEA (sin teams).

Licencia	Gasto por uso mensual
Microsoft Office	35,70€
ASP .NET Core	0€

Angular	0€
Total	**35,70€**

TABLA 11. COSTE DE LICENCIAS MENSUAL

Para el desarrollo y conteo de horas también se usará Clockify, el cual también es gratuito y no se necesitará pagar la licencia.

Evento	Duración (semanas)	Coste total
Preparación del proyecto	4	35,70€
Sprint 1	3	26,80€
Sprint 2	3	26,80€
Sprint 3	3	26,80€
Preparación	2	17,85€
Total	**15**	**133,95€**

TABLA 12. COSTES DE LICENCIAS TOTALES

4.4 Coste del material de trabajo

Aquí se describen los gastos de coste de material de trabajo adquiridos durante el periodo de desarrollo, debido a que su vida útil será de más de 15 semanas de vida, se considerará las amortizaciones de los equipos y materiales adquiridos durante el periodo de desarrollo durante un año de vida.

Item	Precio/unidad
Ordenador portátil	600€

TABLA 13. PRECIO MATERIAL DE TRABAJO

Amortización de los equipos	
Suma total	Periodo de amortización
600	1 año
Amortización semanal	Amortización mensual
12,50€	50€

TABLA 14. AMORTIZACIÓN DE LOS EQUIPOS

Amortización durante el desarrollo	
Periodo de desarrollo	15 semanas
Amortización semanal	12,50€
Total	**187,50€**

TABLA 15. AMORTIZACIÓN DURANTE EL DESARROLLO

Teniendo en cuenta los cálculos finales, en casi 4 meses se va a amortizar unos 187,50€, todo con el I.V.A incluido.

4.5 Desglose total de los costes de desarrollo

En este subapartado están resumidos todos los costes asociados al periodo de desarrollo del producto.

Costes totales del desarrollo (sin reservas de contingencia)	
Costes salariales totales	4.677,44€
Costes derivados totales	225€
Costes de licencia totales	133,95€
Costes de material total	187,50€
Total	**5.223,89€**

TABLA 16. COSTES TOTALES DEL DESARROLLO

4.6 Reservas de contingencia

Aquí se definirán las reservas de contingencia establecidas para el proyecto, que serán de un 10% del coste total (sin contar la propia reserva).

Reservas de contingencia	
Porcentaje de contingencia	10%
Presupuesto total	5.223,89€
Total	**522,39€**

TABLA 17. RESERVAS DE CONTINGENCIA

Si se le suma la reserva de contingencia al coste, el precio total del proyecto sería de: **5.746,28€.**

5. Evolución de la aplicación

5.1 Estudio del arte

En este apartado se va a hacer un estudio del arte con las 7 tecnologías más usadas actualmente en las empresas.

5.1.1 Tecnologías de Front-End

5.1.1.1 React

- **Descripción:**

  React es una biblioteca de JavaScript diseñada para construir interfaces de usuario dinámicas y basadas en componentes. Su enfoque principal es crear aplicaciones web rápidas y eficientes mediante la construcción de interfaces reutilizables y componibles. Se enfoca en la vista dentro del patrón MVC (Modelo – Vista – Controlador).

- **Características:**
  - **Componentes reutilizables**: React organiza las interfaces de usuario en componentes independientes y reutilizables, lo que facilita la gestión y el mantenimiento del código.
  - **Virtual DOM**: React utiliza un sistema llamado Virtual DOM, es más eficiente que el DOM real y solo actualiza las partes necesarias, en comparación con ambas.
  - **Amplio ecosistema de herramientas y bibliotecas**: con herramientas que permite al usuario personalizas sus aplicaciones.
- **Ventajas:**
  - **Flexibilidad:** React no impone una arquitectura específica, lo que permite integrarse fácilmente con otras bibliotecas o frameworks.
  - **Gran comunidad:** cuenta con una enorme comunidad de desarrolladores activos y soporte constante de Facebook, con una gran capacidad de recursos, tutoriales, paquetes de terceros y rápido ciclo de desarrollo.
  - **Buen rendimiento:** el uso de Virtual DOM y el enfoque en la optimización de la actualización de la interfaz, hace que sea muy eficiente en la creación de interfaces.
- **Desventajas:**

- **Curva de aprendizaje:** El ecosistema que rodea a React es agobiante para los principiantes.
- **Rendimiento en actualizaciones grandes**: pueden surgir desafíos de rendimiento si no se implementan correctamente optimizaciones adicionales.

5.1.1.2 Angular

- **Descripción:**

  Angular es un framework de desarrollo de aplicaciones web de código abierto basado en TypeScript, que incluye los necesario para el desarrollo de aplicaciones web con una estructura robusta y manejo de la enrutación.

- **Características:**
  - **Estructura MVC**: este patrón se basa en “Modelo” que representa los datos de la aplicación, “Vista” es la interfaz de usuario y “Controlador” es la lógica que conecta la vista y el modelo.
  - **Inyección de dependencias:** facilita la gestión y reutilización de servicios en diferentes partes de la aplicación.
  - **Herramientas integradas como Angular CLI**: cuenta con una línea de comandos llamada Angular CLI, que simplifica tareas comunes como la creación de proyectos, la generación de componentes, la compilación, pruebas y la optimización.
- **Ventajas:**
  - **Estructurado**: proporciona una arquitectura definida desde el principio, lo que favorece un desarrollo más organizado. Su estructura modular facilita la organización del código y permite dividir grandes aplicaciones en componentes reutilizables.
  - Soporte para aplicaciones grandes: su arquitectura robusta y las herramientas integradas lo hacen ideal para desarrollar aplicaciones de grandes dimensiones que requieren una estructura sólida y mantenimiento prolongado.
  - **Dos vías de enlace de datos:** reduce la necesidad de escribir código manual para sincronizar la interfaz de usuario y el modelo, lo que simplifica el desarrollo en aplicaciones que requieren una alta interactividad.

- **Desventajas:**
  - **Curva de aprendizaje empinada**: es un framework completo con una cantidad significativa de conceptos y herramientas que aprender, como RxJS para programación reactiva, la inyección de dependencias, entre otros.
  - **Complejo:** posee una gran complejidad para aplicaciones pequeñas o simples.

5.1.1.3 Vue.js

- **Descripción:**

  Framework progresivo de JavaScript para construir interfaces de usuario, con un enfoque sencillo y flexible que facilita la integración en proyectas y sistemas.
- **Características:**
  - **Integración gradual**: puede ser adoptado de forma incremental, significa que se puede utilizar Vue.js en una pequeña parte de una página existente o en toda la aplicación.
  - **Reactividad:** permite que los cambios en los datos se reflejen automáticamente en la interfaz de usuario.
  - **Componentes:** se basa en una arquitectura de componentes, que permite a los desarrolladores dividir la aplicación en piezas modulares y reutilizables.
- **Ventajas:**
  - **Fácil de aprender**: se destaca su simplicidad y baja curva de aprendizaje, es un sistema de aprendizaje que utiliza HTML, CSS y JavaScript.
  - **Flexible**: se puede usar para mejorar una parte específica de una página web, ya que no impone una estructura rígida.
  - **Buena documentación**: posee una documentación organizada y compleja, con ejemplos claros y detallados, lo que facilita su aprendizaje.
- **Desventajas:**
  - Menos recursos y herramientas en comparación con React y Angular.
  - Puede ser complejo para proyectos pequeños.

5.1.1.4 Svelte

- **Descripción:**

  Framework para construir interfaces de usuario que compila eficientemente, Svelte compila el código directamente en JavaScript nativo durante el tiempo de desarrollo, generando una código eficiente y optimizado que interactúa con el DOM.

- **Características:**
  - **Compilación en tiempo de desarrollo**: significa que cuando se despliega la aplicación, no hay framework que se ejecuten el navegador, en su lugar, el navegador simplemente ejecuta JavaScript nativo optimizado.
  - **Sin virtual DOM**: Actualiza el DOM real de forma directa, lo que reduce el coste en términos de memoria y procesamiento, proporcionando un rendimiento más eficiente.
  - Código más simple: posee una sintaxis fácil de entender, eliminando la necesidad de escribir mucho código repetitivo o patrones complejos.
- **Ventajas:**
  - **Rendimiento muy alto**: aporta un rendimiento más rápido y eficiente. Se beneficia de tiempos de carga más veloces y un menor consumo de recurso.
  - **Menos código**: concede a los desarrolladores escribir menos códigos para lograr la misma funcionalidad que otros framework. Su sencilla sintaxis disminuye la necesidad de utilizar bibliotecas auxiliares.
- **Desventajas:**
  - Comunidad más pequeña: significa que hay menos recursos, biblioteca y extensiones.
  - La herramientas oficiales y paquetes de terceros son limitados.

5.1.1.5 Ember

- **Descripción:**

  Framework para construir aplicaciones web ambiciosas, parecido a Vue.js pero con más dificultad y complejidad.

- **Características:**
  - **Convención sobre configuración:** trae configuraciones predeterminadas que permiten a los desarrolladores concentrarse en la lógica.
  - **Herramientas de desarrollo integradas:** tiene herramientas útiles que permiten una mejora de productividad del desarrollador.
  - **Enrutamiento robusto:** facilitando la gestión de rutas dentro de la aplicación.
- **Ventajas:**
  - Alta productividad, ofrece plantillas y componentes que permiten un desarrollo rápido.
  - Estructura sólida, con un enfoque estructurado permite aplicaciones bien organizadas y mantenibles
- **Desventajas:** Curva de aprendizaje empinada, menos flexible.
  - Curva de aprendizaje empinada, especialmente para aquellos que no están familiarizados con el desarrollo.
  - Menos flexible, al tener una estructura sólida puede generar desventaja en proyectos que requieran algo más ágil.

5.1.1.6 Backbone.js

- **Descripción:**

  Biblioteca de JavaScript para estructurar aplicaciones web con modelos y vistas. Su principal objetivo es proporcionar una forma sencilla de estructurar el código.
- **Características:**
  - **Ligero:** la biblioteca principal es de tamaño pequeño, lo que es una buena opción para desarrolladores que quieren mantener un alto rendimiento y bajo peso del código.
  - **MVP (Model-View-Presenter):** basado en el “Modelo” que representa los datos de la aplicación; “Vistas” muestra los datos visuales; “Colecciones” conjunto de modelos que pueden manejar listas de datos y por último, “Routers” que permite la navegación.
  - **Sincronización con servidor RESTful**: cuenta con la integración de servicios RESTful, facilitando la sincronización entre los Modelos y un Back-end basado en APIs RESTful.

- **Ventajas:**
  - **Ligero:** posee una biblioteca central muy pequeña, ideal para proyectos que requieren un rendimiento rápido.
  - **Flexible:** permite a los desarrolladores adoptar solo las partes de la biblioteca que necesitan, integrando otras herramientas según sea necesario.
- **Desventajas:**
  - Requiere más configuración: Backbone.js es minimalista y requiere que el desarrollador configure manualmente muchos aspectos de la aplicación.
  - Menos soporte oficial: posee un ecosistema menos desarrollado, no tiene tantas bibliotecas oficiales o soportes, lo que hace que sea más difícil encontrar soluciones o actualizaciones.

5.1.1.7 Preact

- **Descripción:**

  Alternativa ligera a React, diseñada para ofrecer prácticamente la misma funcionalidad, pero con un tamaño de archivo significativamente más pequeño.
- **Características:**
  - API similar a React: significa que los desarrolladores que ya están familiarizados con React pueden cambiar a Preact sin prácticamente cambiar su código.
  - Tamaño muy pequeño y eficiente: esto le permite ser más eficiente en términos de rendimiento y uso de recursos.
  - Virtual DOM: se utiliza para gestionar las actualizaciones de la interfaz de usuario.
- **Ventajas:**
  - **Muy ligero**: su pequeño tamaño permite cargas más rápidas en sitios web y aplicaciones, lo que mejora la experiencia del usuario.
  - **Un gran rendimiento**: al ser minimalista y eficiente permite actualizaciones rápidas de la interfaz sin afectar la interactividad o fluidez.

- **Desventajas:**
  - **Menos características integradas:** no incluye algunas características avanzadas por lo que los desarrolladores pueden verse limitados.
  - **Menos comunidad:** posee una comunidad más pequeña, lo que significa que hay menos recursos.
  - **Menos información**: ha sido adoptado por un número creciente de desarrolladores, aún no ha alcanzado un nivel de madurez suficiente.

5.1.1.8 Tabla comparativa sobre estudios de Front-End

A continuación, se observa una tabla comparativa de las características de las tecnologías de Front-End

FRONTEND	React	Angular	Vue.js	Svelte	Ember.js	Backbone.js	Preact
Componentes reutilizables	✓	✓	✓	✓	✓	✗	✓
Amplio ecosistema de herramientas y bibliotecas	✓	✓	✓	✗	✓	✗	✗
Estructura MVC	✗	✓	✗	✗	✗	✓	✗
Inyección de dependencias	✗	✓	✗	✗	✗	✗	✗
Herramientas integradas	✗	✓	✓	✓	✓	✗	✗
Enrutamiento robusto	✓	✓	✓	✓	✓	✗	✗
Productividad alta	✓	✓	✓	✓	✓	✗	✓
Flexibilidad	✓	✗	✓	✓	✗	✓	✓
Ligero	✓	✗	✓	✓	✗	✓	✓
Soporte para aplicaciones grandes	✓	✓	✓	✗	✓	✗	✗
Dos vías de enlace de datos	✗	✓	✓	✓	✗	✗	✗
Comunidad	✓	✓	✓	✗	✗	✗	✗
Buen rendimiento	✓	✓	✓	✓	✓	✓	✓
Curva de aprendizaje empinada	✓	✓	✗	✗	✓	✗	✗

TABLA 18. TABLA COMPARATIVA FRONT-END

5.1.2 Tecnologías de Back-End

5.1.2.1 Node.js

- **Descripción:**

  Es un entorno de ejecución que ejecuta JavaScript del lado del servidor, por lo que permite su utilización en el entorno Back-End, proporciona un gran rendimiento pensado para aplicaciones escalables.

- **Características:**
  - **Event-driven:** modelo de programación basada en eventos, no ejecuta el código de forma secuencial si no que emite eventos al completarse una aplicación.
  - **Non-blocking I/O:** fundamental para su eficiencia, procesa las operaciones de manera asíncrona por lo que es adecuado para aplicaciones en tiempo real.
  - **Gran ecosistema (NPM):** manager de control de paquetes en los que se encuentran módulos capaces de resolver gran cantidad de necesidades.
- **Ventajas:**
  - Trabaja de forma asíncrona, es muy eficiente por lo que es capaz de manejar múltiples conexiones.
  - Escalabilidad, gracias a la capacidad de eficiencia permite escalar verticalmente, añadiendo más potencia a sus servidores u horizontalmente, añadiendo más servidores.
  - Simple curva de aprendizaje, ya que utiliza JavaScript para todo.
- **Desventajas:**
  - No es adecuado para tareas intensivas de CPU, como utiliza un único hilo, por lo tanto, con una tarea intensiva el CPU bloquearía el hilo principal.
  - Callback hell, aunque con una buena organización de código no debería pasar, es devolver datos sin que se hayan obtenido todavía.

5.1.2.2 Django

- **Descripción:**

  Framework que permite crear páginas web rápidamente de forma eficiente, enfocado en MVC con funcionalidades que aceleran el desarrollo de la aplicación.

- **Características:**
  - **Estructura MVC:** la capa “Modelo” se encarga de representar los datos, la “Vista” maneja la lógica en la presentación y el controlador se utiliza para crear las paginas html
  - **ORM incorporado:** esto facilita las operaciones con las bases de datos sin escribir código SQL, Django es compatible con múltiples bases de datos como PostgreSQL, MySQL, SQLite y Oracle.
  - **Herramientas de administración:** tiene una interfaz de administración integrada, que permite una gestión de las tablas de la base de datos.
- **Ventajas:**
  - Seguridad, gestionando la autenticación de usuarios y preparado para ataques como SQL Injection
  - Desarrollo rápido, con un conjunto de herramientas listas para usar, reutilización de código, está diseñado para ayudar a los desarrolladores a construir aplicaciones rápidamente.
  - API REST integrada, proporciona herramientas para construir APIs Restful sencillas y escalables
- **Desventajas:** Pesado, menos flexible.
  - Pesado, con una gran curva de aprendizaje debido a sus funcionalidades, puede hacerse pesado para un proyecto simple.
  - Menos flexible, Django tiene una estructura rígida que puede resultar restrictiva para quienes quieran más flexibilidad.

5.1.2.3 Ruby on Rails

- **Descripción:**

  Framework web basado en Ruby siguiendo el principio de “No te repitas” se utiliza para hacer aplicaciones web enfocadas en la productividad del desarrollador.

- **Características:**
  - **Convención sobre configuración:** esto permite a los desarrolladores enfocarse más en el código que en la configuración de la aplicación.
  - **Herramientas integradas:** como autenticación de usuario, validación de formularios generadores de código…
  - **ORM (ActiveRecord):** permite a los desarrolladores interactuar fácilmente con bases de datos relacionales.
- **Ventajas:**
  - Productividad, gracias a las herramientas integradas permite a los desarrolladores ser altamente productivos, ideal para startups.
  - Orientación a prototipos rápidos, dado que por su velocidad de desarrollo es útil para MVPs y prototipos.
- **Desventajas:** Rendimiento, curva de aprendizaje elevada.
  - Rendimiento, rendimiento inferior al de otros frameworks, especialmente con aplicaciones de altas peticiones.
  - Curva de aprendizaje elevada, especialmente para desarrolladores que no conocen Ruby.

5.1.2.4 Spring Boot

- **Descripción:**

  Framework que facilita el desarrollo de aplicaciones Java, principalmente enfocadas a aplicaciones de microservicios robustas con una configuración automática simplificando su configuración.
- **Características:**
  - **Microservicios:** divide la aplicación en pequeños servicios independientes pudiendo ser reutilizables.
  - **Configuración automática:** detecta automáticamente los componentes y las dependencias necesarias por lo que gran parte de configuración manual es evitada.
  - **Seguridad incorporada:** cuenta con Spring Security una herramienta para gestionar la seguridad de las aplicaciones con autenticación, cifrado y otras medidas de seguridad.

- **Ventajas:**
  - Potente, diseñado para aplicaciones grandes con una arquitectura sólida, este ecosistema permite desarrollar cualquier aplicación.
  - Escalable, gracias a los microservicios es capaz de integrar soluciones de la nube, es ideal para la escalabilidad horizontal.
- **Desventajas:** Configuración compleja, pesado.
  - Configuración compleja, al personalizar aspectos avanzados de una aplicación la configuración puede ser compleja, como la integración de varios microservicios.
  - Pesado, con la variedad de bibliotecas y características de Sprint Boot, las aplicaciones suelen ser pesadas en términos de recursos.

5.1.2.5 Express.js

- **Descripción:**
  Framework minimalista y ligero que sirve para construir aplicaciones web y APIs en Node.js,
- **Características:**
  - **Middleware:** es una función que se ejecuta en el ciclo de vida de una petición HTTP, esta puede modificar la petición.
  - **Ruteo sencillo:** capacidad de manejar diferentes solicitudes HTTP con facilidad, puede definir tanto rutas estáticas como dinámicas.
  - **Gran comunidad:** uno de los más usados para Node.js que ha generado una comunidad de desarrolladores con una gran colección de recursos.
- **Ventajas:**
  - Extremadamente ligero, proporciona herramientas esenciales de ruteos y middleware.
  - Flexible, no impone una estructura rígida por lo que se adapta a una gran cantidad de proyectos.
- **Desventajas:**

- Requiere más configuración, necesita mayor configuración manual ya que no lo trae incorporado.
- Menos estructurado, puede ser una desventaja cuando se trata de equipos de desarrollo grandes.

5.1.2.6 Flask

- **Descripción:**
  Microframework web utilizado en Python, simple, minimalista y fácil de usar. A diferencia de Django, Flask trae lo esencial para el desarrollo de páginas web.
- **Características:**
  - **Minimalista:** incluye herramientas y funcionalidades básicas para el desarrollo, esto lo hace extremadamente ligero.
  - **Extensible:** los desarrolladores pueden incluir funcionalidades externas mediante bibliotecas según las necesidades.
  - **WSGI:** especificación estándar para servidores y aplicaciones en Python.
- **Ventajas:**
  - Ligero, tanto en términos de complejidad y de manejo de recursos, permite construir aplicaciones de manera rápida y sin sobrecarga.
  - Fácil de aprender, más que otros frameworks más completos como Django especialmente para aquellos que ya conocen Python
- **Desventajas:** No tiene muchas herramientas integradas.
  - La falta de herramientas hace que desarrolladores que necesiten más funciones tengan que implementarlas externamente y depender de extensiones.
  - No dispone de una estructura específica para proyectos, puede ser una desventaja para proyectos grandes.

5.1.2.7 ASP .NET Core

- **Descripción:**
  Framework de código abierto que permite crear aplicaciones web modernas utilizando el lenguaje C#, diseñado para ser ligero, modular y completamente Cross-platform. Es ideal para aplicaciones web con APIs Restful.

- **Características:**
  - **Cross-platform:** las aplicaciones pueden ejecutarse tanto en Windows, como en Linux y macOS.
  - **Inyección de dependencias:** para ello utiliza paquetes Nugget facilitando el uso de buenas prácticas.
  - **Buen rendimiento:** es uno de los frameworks más rápidos actualmente, con arquitectura modular y uso de Kestrel, un navegador web ultrarápido.
- **Ventajas:**
  - Integración con el ecosistema .NET, con una amplia variedad de herramientas y bibliotecas disponibles como Entity Framework Core.
  - Actualizaciones continuas y soporte de Microsoft
  - Rendimiento, con el uso de Kestrel, se obtiene un manejo de peticiones HTTP con tiempos de respuestas rapidos.
- **Desventajas:**
  - Curva de aprendizaje, a pesar de sus facilidades tiene una curva de aprendizaje alta, sobre todo para personas que no han trabajado con C#.
  - Complejidad para principiantes, puede ser más complejo que frameworks mas minimalistas como Flask o Express.js.

5.1.2.8 Tabla comparativa sobre tecnologías Back-End

A continuación, se va a ver como en el apartado anterior, una tabla comparativa sobre las tecnologías del Back-End.

BACKEND	Node.js	Django	Ruby on Rails	Spring Boot	Express.js	Flask	ASP.NET Core
Estructura MVC	✗	✓	✓	✓	✗	✗	✓
ORM incorporado	✗	✓	✓	✗	✗	✗	✓
Herramientas de administración	✗	✓	✓	✓	✗	✗	✓
Microservicios	✗	✗	✗	✓	✗	✗	✓
Configuración automática	✗	✗	✗	✓	✗	✗	✓
Seguridad incorporada	✗	✓	✓	✓	✗	✗	✓
Ruteo sencillo	✗	✗	✗	✗	✓	✗	✗
Extensible	✓	✗	✗	✗	✓	✓	✓
Cross-platform	✓	✗	✗	✓	✓	✓	✓
Inyección de dependencias	✗	✗	✗	✓	✗	✗	✓
Buen rendimiento	✓	✓	✗	✓	✓	✓	✓
Ligero	✓	✗	✗	✗	✓	✓	✗
Curva de aprendizaje	✓	✓	✓	✓	✓	✓	✓
Trabajo de forma asíncrona	✓	✗	✗	✗	✓	✗	✓

Gran comunidad	✓	✓	✓	✓	✓	✓	✓

TABLA 19. TABLA COMPARATIVA BACK-END

5.2 Fase de diseño del producto

Al ser una aplicación que va a tener un uso real con diferentes personas (tanto como aplicación de pruebas en el ámbito bancario, como aplicación escolar para el profesor Juan Antonio Andrés Lalueza, en sus clases de economía) se ha tenido una preocupación por la estética y el nivel de accesibilidad.

Para realizar la fase de diseño se ha utilizado “Bootstrap”, en conjunto con el material de “Icons de Google”, pues facilitan la representación del diseño y la funcionalidad de la aplicación para los usuarios.

Para la creación del diseño, la interfaz de usuario conocida con las iniciales UI y la experiencia de usuario reconocida como UX, se han empleado las siguientes características:

- **Planteamiento y maquetación:** En lo referente a los diferentes apartados de la aplicación, se puede destacar los bocetos o mockups, los cuales han dado un esquema de diseño a la misma, la herramienta con la cual se ha trabajado para la realización de dichos mockups es una herramienta online llamada Figma. Haciendo referencia al nombre, se ha formado haciendo un juego de palabras, cambiando letras del nombre de uno de los bancos más importantes en España.

- **Diseño del concepto**: Para el diseño se ha optado por una alternativa sencilla, donde los usuarios podrán reconocer de manera fácil cada uno de los conceptos y funciones que componen la aplicación. Para una mejor visualización de la aplicación, previamente se ha creado mockups de su diseño, estos se mostrarán posteriormente.

  Por último, en esta fase han ajustado las diferentes paletas de colores para crear un aspecto visual agradable y observar con antelación el aspecto de la versión final.

- **Prototipado y diseño detallado**: En este apartado, se han realizado todos los cambios definitivos que han definido el aspecto de la versión final de la aplicación y el funcionamiento de la misma.

En esta fase se ha concretado también los iconos e imágenes usados a lo largo de las vistas de la aplicación. Estos recursos visuales externos se han llevado a cabo detallando su código tal y como se especifica en la página donde se han obtenido.
Las páginas principales de donde se han obtenido estos recursos, como se ha mencionado anteriormente son:

- **Material Icons de Google:** Estos son los iconos que se han usado en la aplicación provenientes de la API oficial de fuentes de Google.

- **Bootstrap:** Esto es una página a la que se hace referencia para obtener un CSS (Cascading Style Sheets) externo el cual nos ayuda a diseñar y estilar la página de una manera más fácil y eficiente.

En todo el proceso se ha llevado un nivel de armonía en los diferentes elementos de cada vista, introduciendo iconos, placeholders y una gama de colores agradable a la vista del usuario. Aparte se han utilizado imágenes acordes a las vistas de la aplicación.

5.2.1 Mockups y prototipo

A continuación, se muestran cada una de las vistas y funciones que representa la aplicación, a través de mockups. Se va a proceder a hacer una división del prototipado de la aplicación, esta división se compone de tres partes: no logeados, usuario normal y usuario administrador.

Primeramente, se procederá a enseñar la parte de usuarios no logeados, que consta de una vista inicial, en la que se muestra la fase de acceso a la aplicación.

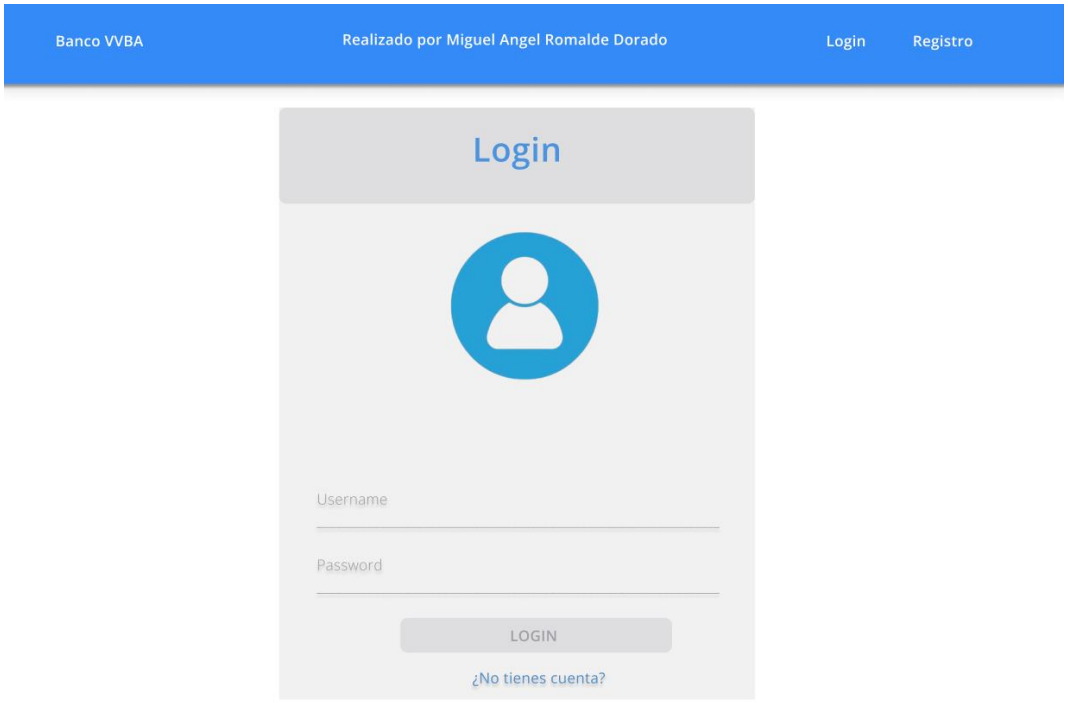

ILUSTRACIÓN 17. LOGIN

Seguidamente, se muestra la fase de registro del usuario, en la cual, se obtienen todos los datos necesarios de la persona para poder acceder correctamente, usando las credenciales del usuario.

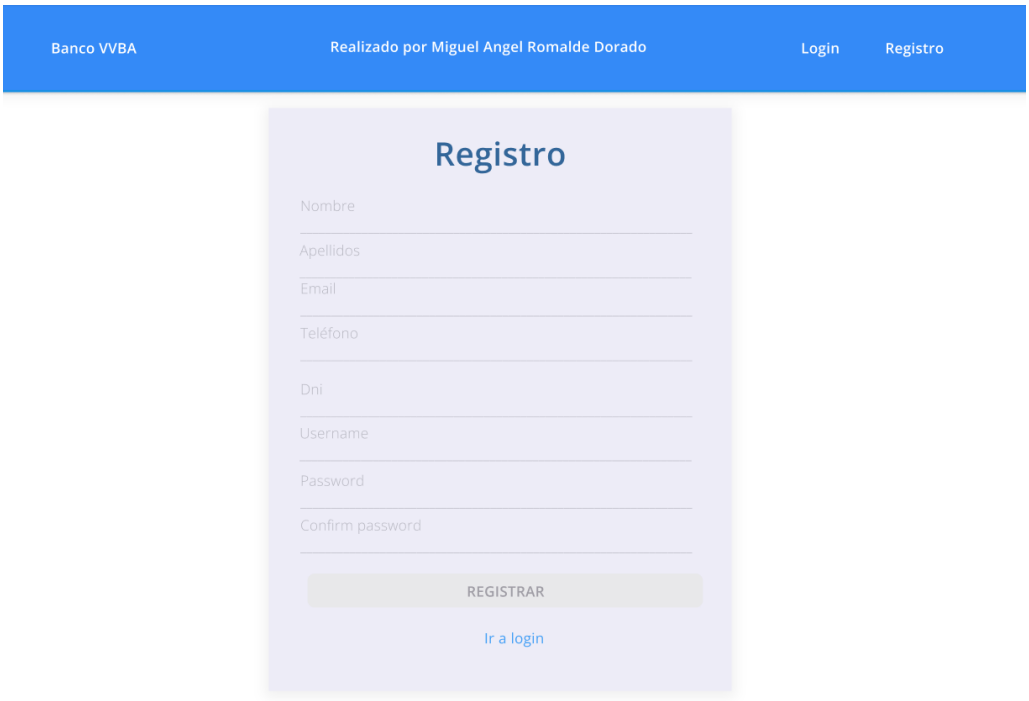

ILUSTRACIÓN 18. REGISTRO

Se entra ahora en la fase de usuarios normales, en la cual se accede una vez que el usuario se ha logeado en la aplicación. Esta primera pantalla hace referencia a los datos de la cuenta del usuario y las operaciones de la misma.

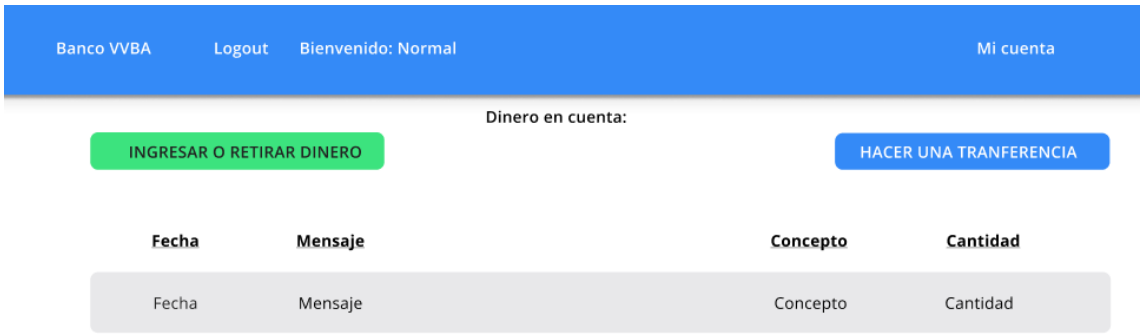

ILUSTRACIÓN 19. DETALLES CUENTA USUARIO.

En la siguiente ilustración se muestra el formulario referente al ingreso o retirada de dinero en el banco. Los usuarios accederán a esta vista pulsando el botón “ingresar o retirar dinero” que se observa en la imagen anterior.

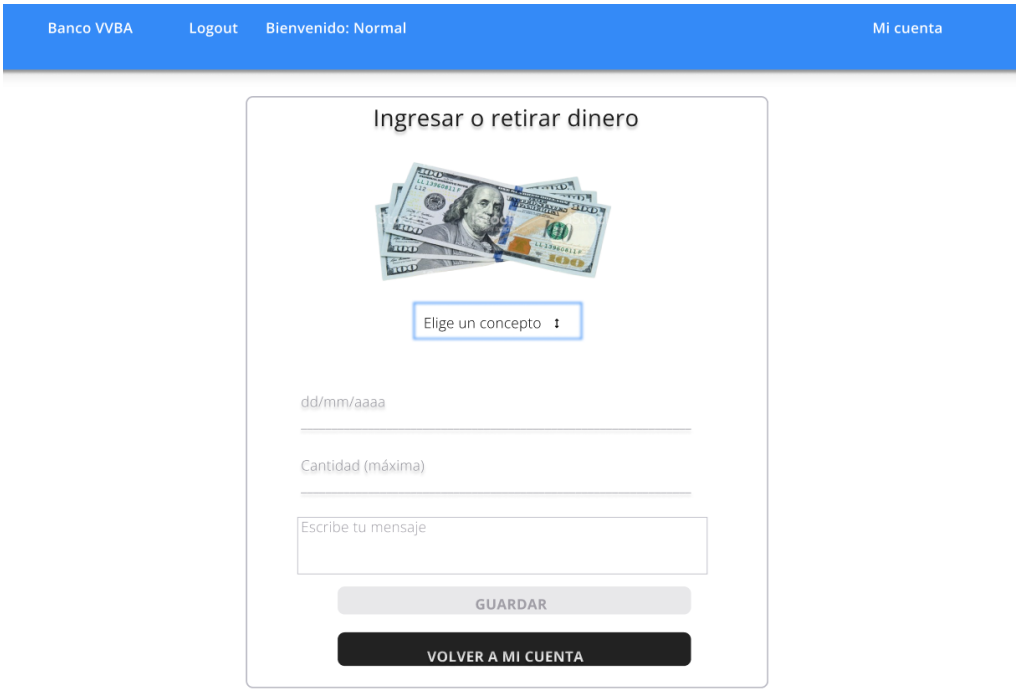

ILUSTRACIÓN 20. INGRESAR O RETIRAR DINERO

Además de ingresar o retirar dinero, los usuarios podrán realizar transferencias de dinero hacia otra cuenta registrada en la aplicación. Los usuarios accederán a esta vista pulsando el botón “hacer una transferencia” que se observa en la ilustración 19.

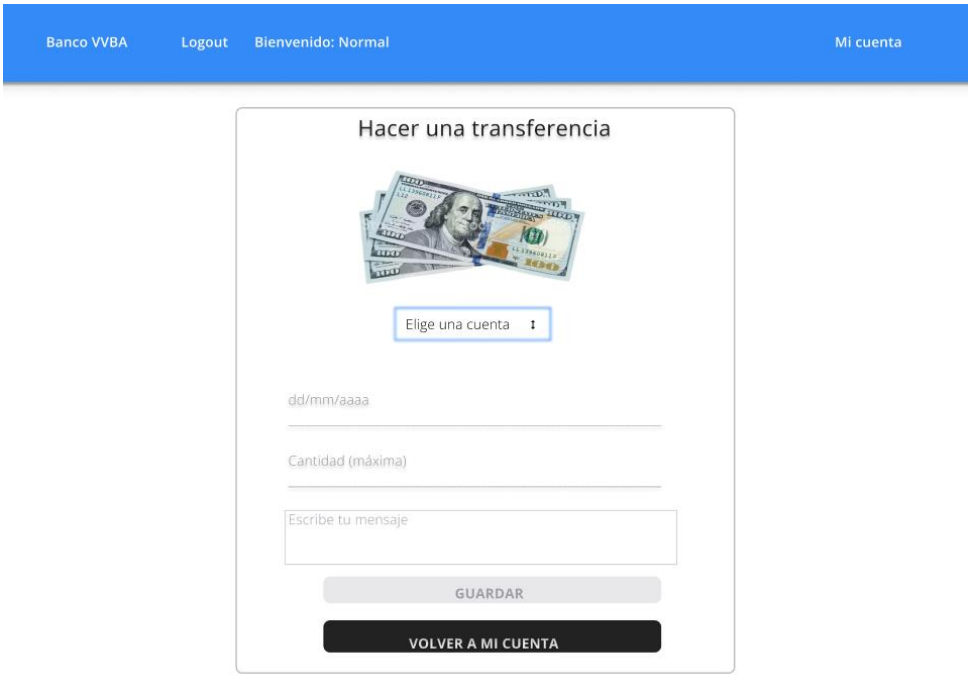

ILUSTRACIÓN 21. TRANSFERENCIAS

Por último, se entra en la fase de usuarios administradores, esta es la mayor fase y consta de 9 vistas, dado que hay algunas parecidas, se pondrá una imagen de ejemplo y se hará referencia a las vistas a las que hace referencia.

Esta es la vista de usuarios, la cual muestra diferentes datos de los usuarios de la aplicación, esta vista es la que, mencionada anteriormente, hace referencia a 5 vistas, ya que la vista de usuarios, cuentas, operaciones, comisiones, y cuentas y comisiones son similares a esta con cambios menores.

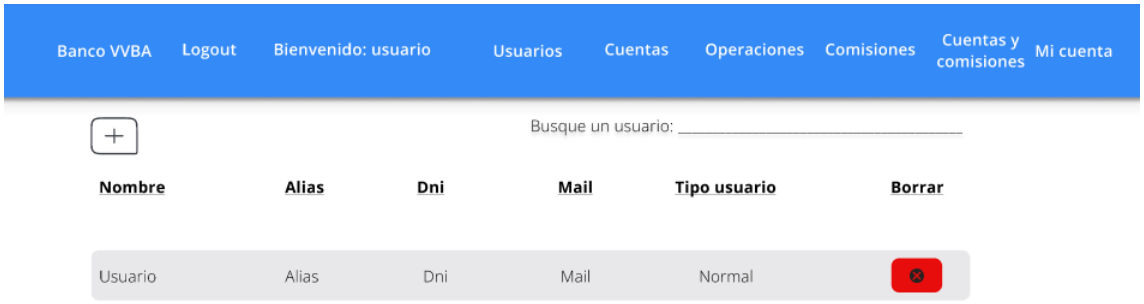

ILUSTRACIÓN 22. LISTA DE USUARIOS

En la siguiente vista se puede observar la creación de un usuario hecha por el usuario administrador, en esta se puede elegir el tipo de usuario en diferencia al registro. Se puede navegar a esta vista se debe pulsar el botón o símbolo “+” de la esquina superior izquierda de la imagen anterior, en la pantalla de usuarios.

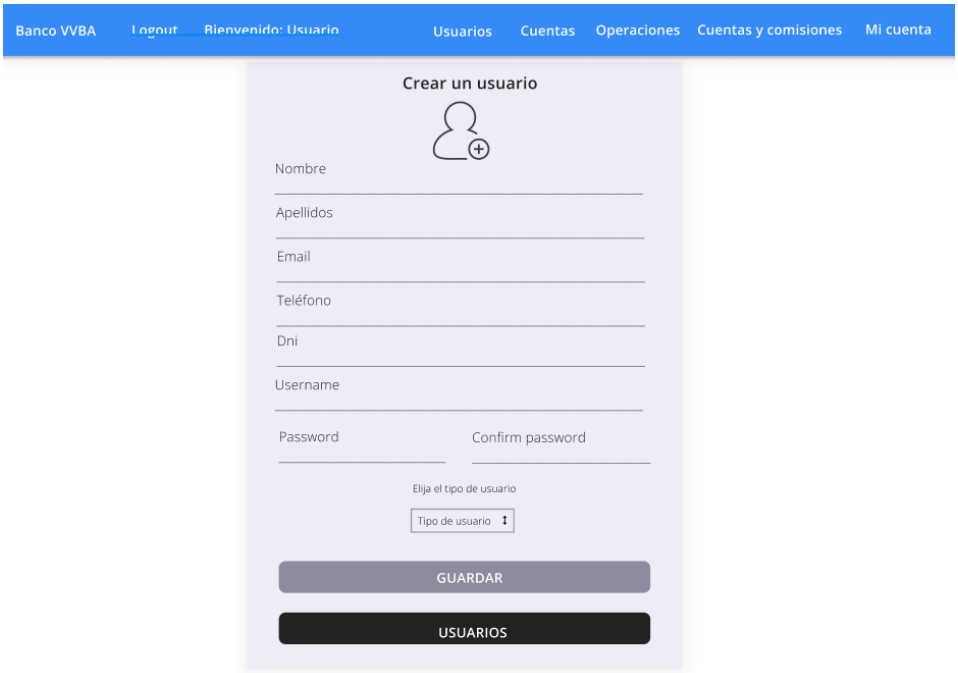

ILUSTRACIÓN 23. CREACIÓN USUARIO

En la posterior vista se puede ver la creación de una operación dentro de una cuenta, se pondrá en el placeholder, la cantidad máxima a ingresar o a retirar de forma dinámica dependiendo de la cuenta elegida.

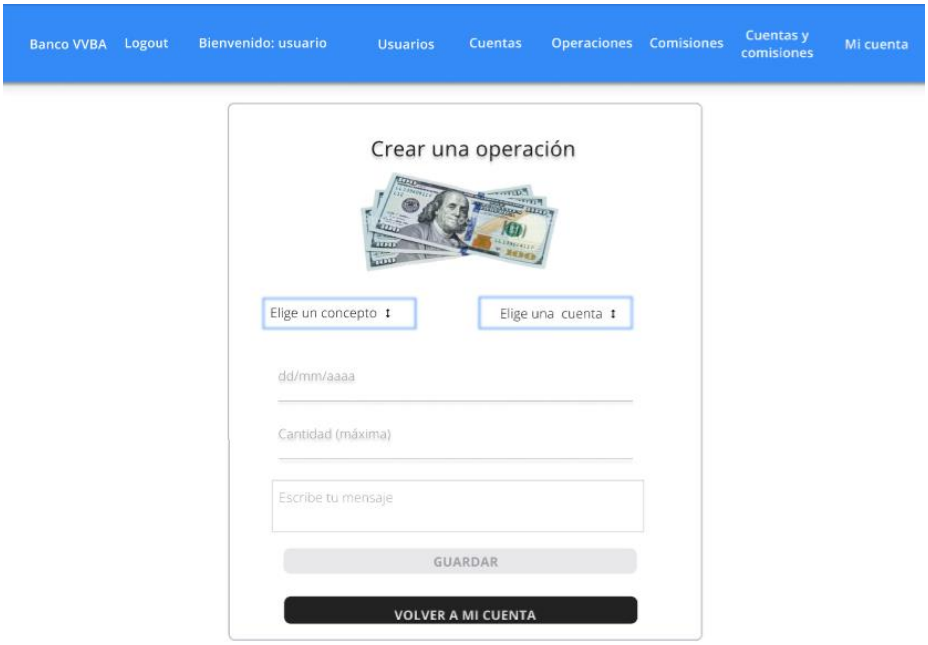

ILUSTRACIÓN 24. CREACIÓN OPERACIÓN

En la siguiente pantalla se puede crear un tipo de comisión dicha comisión podrá ser usada en la aplicación.

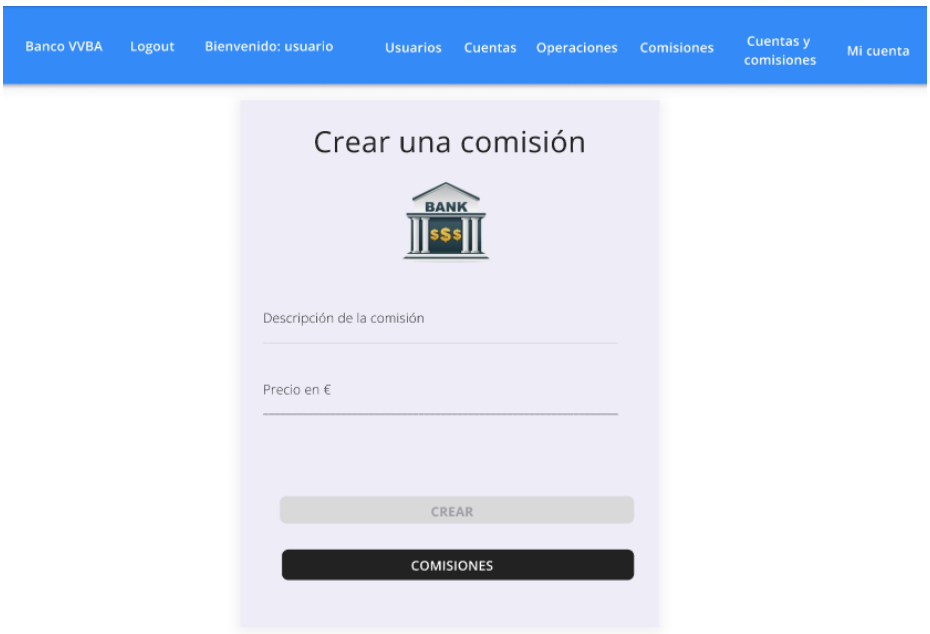

ILUSTRACIÓN 25. CREACIÓN COMISIÓN

Se puede adjuntar una comisión a una cuenta, para ello se debe elegir el tipo de cuenta y el tipo de la comisión que se vaya a añadir.

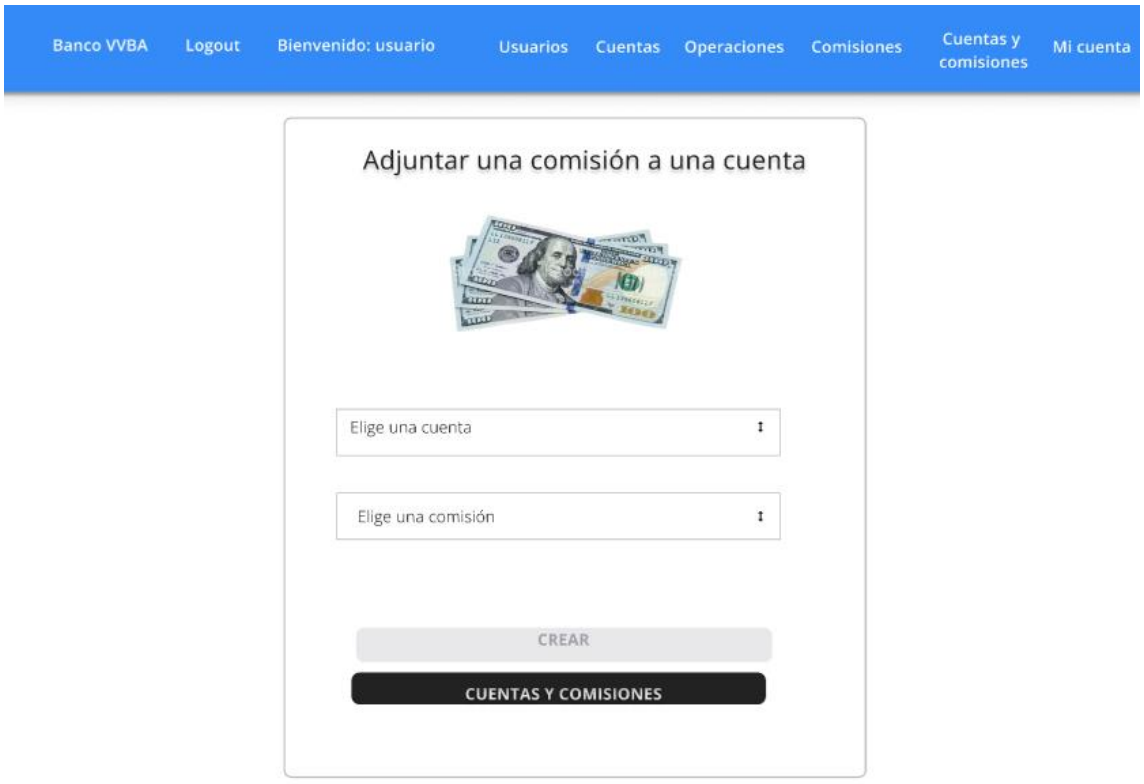

ILUSTRACIÓN 26. CREACIÓN COMISIÓN EN CUENTA

Por último, el propio usuario administrador que esté logeado, también es un usuario de la aplicación, por lo que, también dispone de su cuenta y es capaz de hacer las mismas operaciones que un usuario normal.

ILUSTRACIÓN 27. CUENTA ADMINISTRADOR

5.3 Fase de implementación

Dentro de la fase de implementación se procederá a explicar todos los subapartados que se han llevado a cabo.

5.3.1 Herramientas y servicios.

Para un correcto desarrollo de la aplicación se han seleccionado las siguientes herramientas para desarrollo y gestión.

- Git: esta herramienta se ha utilizado para un control del versionado del proyecto en sí. Es una herramienta estandarizada la cual se usará en concreto Github que cuenta con todas las funcionalidades buscadas.
- Visual Studio Code: es el IDE seleccionado para la realización del Front-End de la aplicación, se ha utilizado múltiples veces este IDE en el grado lo cual nos ayuda a tener un trabajo más fácil y eficiente. Además, se dispone de plugins, los cuales nos ayudan al desarrollo y a una fácil comprensión de las carpetas internas del proyecto.
- Visual Studio: se va a utilizar este IDE para la realización del Back-End del proyecto, dado que se quiere usar ASP .NET Core, este es el IDE recomendado para su uso, aunque no se ha usado este IDE en la universidad, si lo he usado anteriormente en proyectos personales por lo que no se gastará tiempo en aprender cómo utilizarlo.
- SQL Server: se ha elegido SQL Server, utilizado con SQL Server Management Studio como forma de acceso. Esta base de datos

relacional permitirá almacenar todos los datos necesarios de la aplicación con un fácil acceso y manejo.

- Postman: esta herramienta nos permite hacer llamadas HTTP a la aplicación.

Todas las herramientas mencionadas anteriormente no han tenido una fase de aprendizaje previa ya que, se conocía su funcionalidad.

En los siguientes dos apartados, se va a explicar cómo ha sido el desarrollo en cada una de las tecnologías utilizadas, su arquitectura, layers, ciclo de vida y estructura de la aplicación.

5.3.2 Desarrollo en ASP .NET Core.

Una vez seleccionada la base para el desarrollo del proyecto, se ha comenzado con una investigación, ya que, aunque ya había una experiencia con el IDE no existía la misma experiencia con la tecnología a utilizar, por ello, primero se investigó como realizar la conexión a la base de datos (DB), como se han de almacenar objetos en dicha DB, su persistencia y cómo usar el patrón MVC (modelo vista controlador) el cual está dirigido ASP .NET Core.

Para ello, primero se explicará que son los paquetes Nugget, esto es un archivo ZIP con extensión .nupkg que contiene código compilado en su interior, con una funcionalidad parecía a Dynamic-Link Library (DLL), este paquete se descarga mediante una herramienta interna de Visual Studio y se descomprime dentro del proyecto, el cual nos permite utilizar las funcionalidades de la librería externa.

Aparte de los paquetes Nugget esenciales de los que se dispondrá cuando se cree un proyecto ASP .NET Core, se necesitarán de otros tres paquetes externos para un correcto funcionamiento de nuestra aplicación, estos tres paquetes son: Swashbuckle.AspNetCore , Microsoft.EntityFrameworkCore.tools y Microsoft.EntityFrameworkCore.SqlServer.

- Swashbuckle.AspNetCore: este paquete Nugget nos proporciona herramientas para poder documentar la API usada mediante Swagger. Esto es una herramienta que nos muestra todos los endpoints con información de cómo funciona, atributos y resultados de respuesta de nuestra API.

- Microsoft.EntityFrameworkCore.tools: es un paquete necesario para poder utilizar el siguiente paquete, sin ello no se podría tener un correcto funcionamiento del paquete.
- Microsoft.EntityFrameworkCore.SqlServer: es el paquete esencial por el cual, con las herramientas que trae este paquete, se podrá conectar fácilmente a la DB que se ha utilizado en el proyecto.

En el siguiente apartado se profundizará más a fondo en la arquitectura de esta tecnología.

5.3.2.1 Arquitectura de ASP .NET Core

La arquitectura seguida por esta tecnología funciona como una aplicación monolítica, ya que, toda la lógica está contenida en un solo proyecto, se compila en una solución e implementa como una unidad.

Contiene todo el comportamiento de la aplicación como se muestra en la siguiente ilustración.

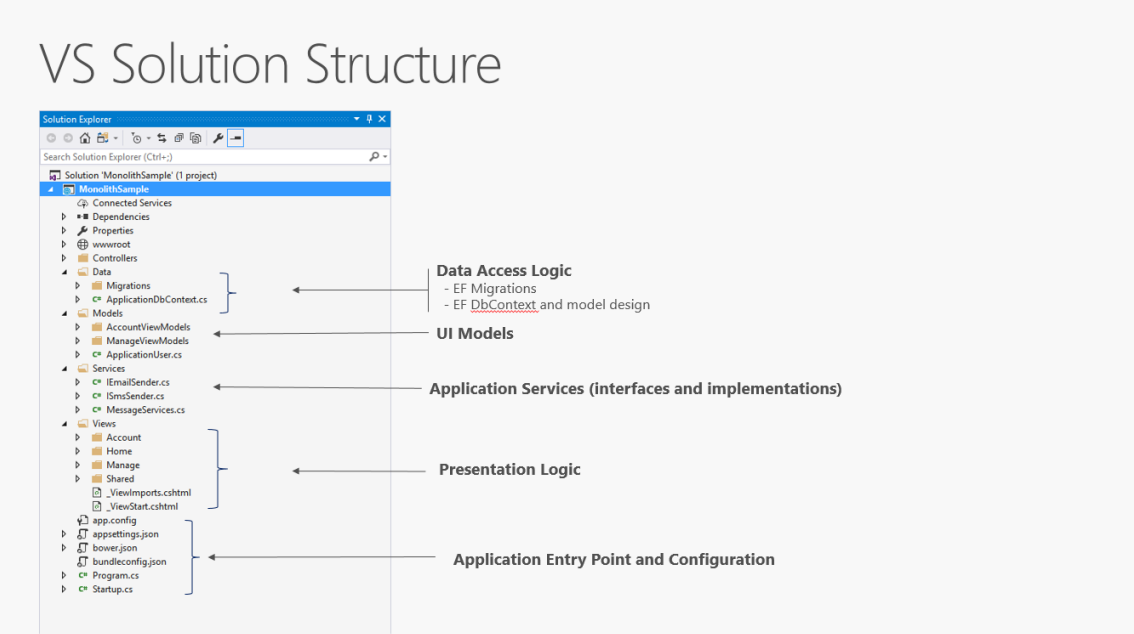

ILUSTRACIÓN 28. ARQUITECTURA ASP .NET CORE

En la ilustración anterior se pueden distinguir los siguientes elementos:

- UI Models: esta parte hace referencia a los modelos de la aplicación, estos modelos una vez se haga una migración de la DB se convertirán en las tablas.
- Data Access Logic: se podrá diferenciar las migraciones de la aplicación y el contexto de la DB, este contexto contiene toda la información y elementos de nuestra DB.

- Application Services: Contiene la implementación de la lógica de la aplicación.
- Entry Point and Configuration: estos son archivos de configuración que se utilizará para configurar el proyecto.

El elemento de Presentation Logic, no se ha explicado ya que en el proyecto se utilizará Angular para la parte del Front-End y no se utilizará esta parte.

5.3.2.2 Layers.

Se va a proceder a explicar los layers que se han utilizado en el proyecto como se ve en la siguiente imagen.

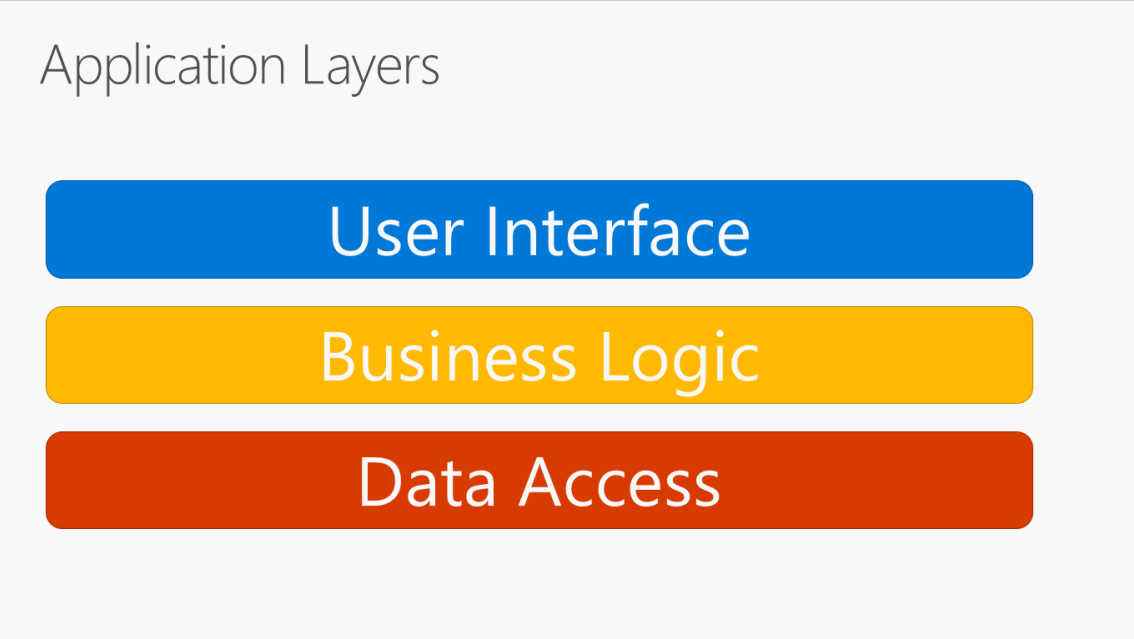

ILUSTRACIÓN 29. LAYERS ASP .NET CORE

En el proyecto se ha utilizado diferentes layers en el funcionamiento de la lógica de negocio, se diferencian tres layers esenciales: Controller, Service y Repository respectivamente.

- Controller: es la primera capa de la aplicación donde llegarán los datos del Front-End, cabe destacar que los datos que viajan entre las dos tecnologías viajan en formato JSON, este layer se encarga de recibir datos y enviárselos al Service.

- Service: es la capa intermedia, tiene una importante función, realizar la lógica (en caso de que sea necesario) dentro del código y traspasar los datos a la última capa, la de Repository.

- Repository: la última capa de la aplicación, esta capa es muy importante, ya que, es la encargada de conectar con la DB, actualizarla y devolver los datos necesarios hacia las capas posteriores.

Estas son las capas en las cuales el proyecto se ha basado, en el siguiente punto se explicará el ciclo de vida de ASP .NET Core.

5.3.2.3 Ciclo de vida

Para comprender la aplicación se va a hacer un ciclo de vida de los datos que son utilizados en el proyecto.

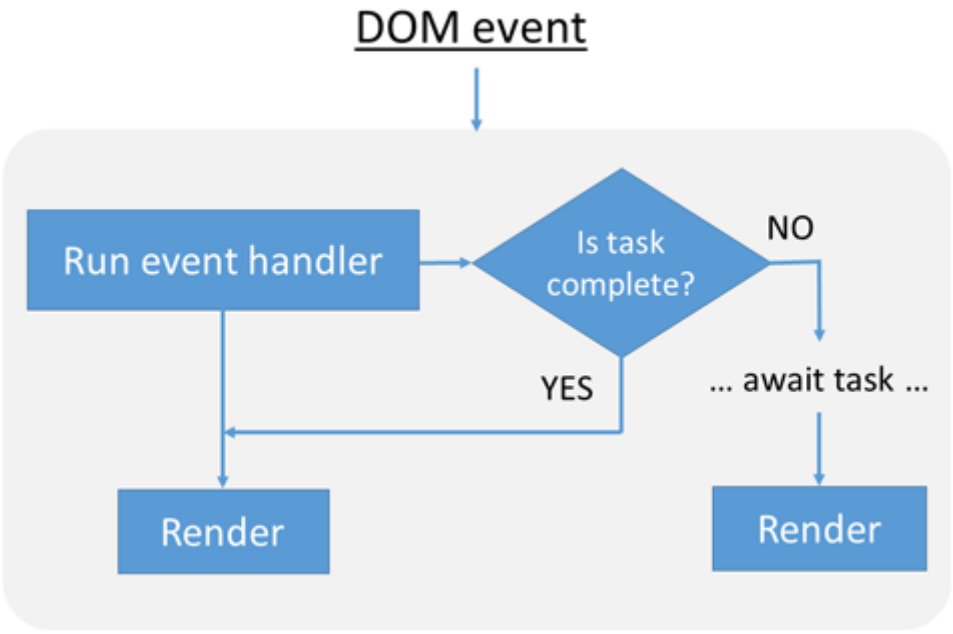

ILUSTRACIÓN 30. CICLO DE VIDA ASP .NET CORE

Como se muestra en la ilustración anterior, ASP .NET Core una vez arrancado, queda a la espera de algún cambio en el DOM, al haber una llamada a nuestra aplicación, se arranca un manejador de eventos, y sigue el proceso de la imagen, al trabajar en formato asíncrono, se tiene que añadir await a las tareas para que devuelva los datos de forma segura, este proceso vuelve a repetirse hasta que para la aplicación.

A continuación, se muestra la estructura del proyecto.

5.3.2.4 Estructuración del proyecto.

La estructuración del proyecto es similar a la ilustración 28, ya que, hay archivos que no pueden ser borrados debido a que son necesarios para el arranque de la aplicación, como se mostrará en la siguiente imagen.

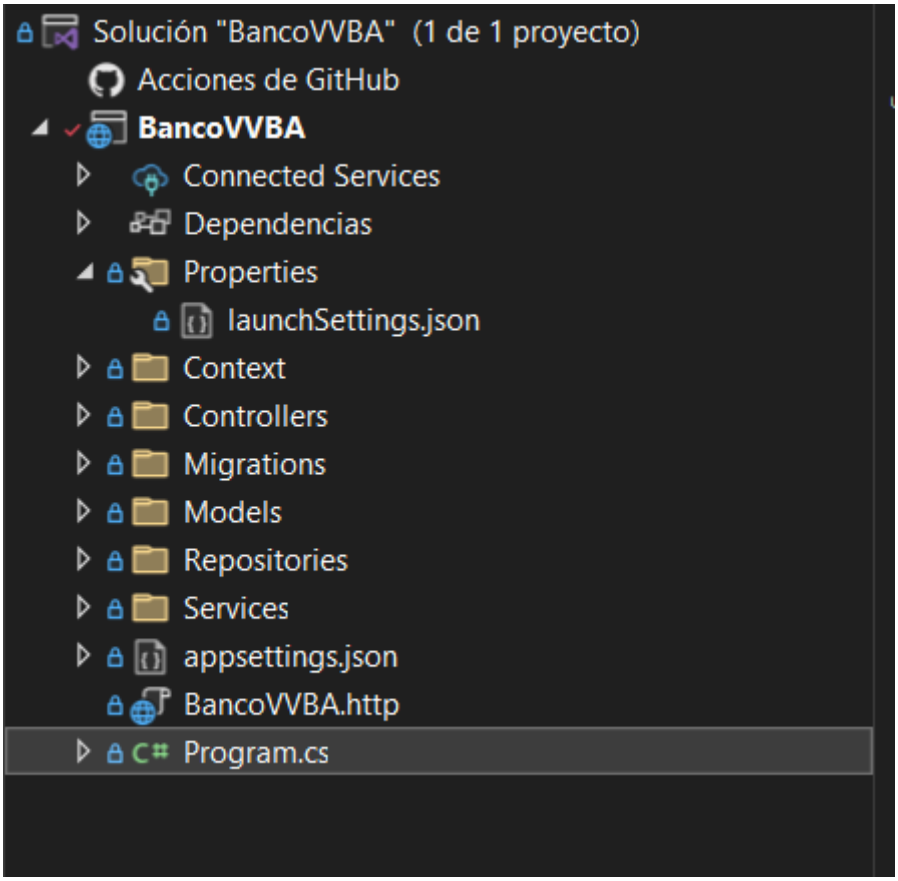

ILUSTRACIÓN 31. ESTRUCTURA PROYECTO ASP .NET CORE

Como se puede observar en la ilustración 31 y como se ha ido comentando anteriormente, el proyecto ha sido dividido en diferentes carpetas.

- Controller, Service y Repository: estos son las capas en las que se ha dividido la lógica de negocio.
- Models: son los modelos, en los que se basa el contexto para crear las tablas de la DB.
- Configuración: los últimos tres archivos son los archivos de configuración de la aplicación en estos esta la conexión a la base de datos y la configuración de arranque de la aplicación.

- Migrations: contiene las migraciones de nuestra DB con los cambios que haya habido en esta.
- Context: contiene el archivo de contexto de la base de datos donde hace referencia a los modelos para crear las tablas de nuestra DB.

Se procederá en el siguiente punto a ver el estado de las pruebas.

5.3.2.5 Pruebas

Debido a la magnitud del proyecto no hay pruebas unitarias de ninguno de los dos tipos, sin embargo, esto no quiere decir que la aplicación no haya sido probada, si no, que las pruebas no han sido automatizadas, pero si se han hecho de forma manual y han sido bastante intensas a lo largo del desarrollo de la aplicación.

Cabe destacar que, aunque se hubieran hecho pruebas automatizadas, la aplicación hubiese tenido que probarse de forma manual de todas formas. También cabe destacar que en la parte del Front-End desarrollado en angular están todos los datos probados para que lleguen en un formato adecuado y de forma para un correcto funcionamiento de la aplicación en la parte del Back-End.

5.3.3 Desarrollo en Angular

Como se ha hablado anteriormente, la parte de la aplicación correspondiente al Front-End ha sido desarrollada en Angular, esta ha sido una tecnología totalmente nueva y se han utilizado varios tutoriales para aprender sobre su uso, concretamente uno interno que nos proporciona la página oficial de Angular llamado “Tour of Heroes”, siguiendo este tutorial se ha obtenido una aproximación bastante buena sobre cómo funciona esta tecnología.

Primero hay que explicar el funcionamiento de esta tecnología, para que no sea muy extenso mucho, se va a explicar por encima como funciona. Primero se crea el proyecto y se crean varias carpetas, el código del proyecto vendrá definido en la carpeta que se llame como nuestro proyecto, una vez ingresado a esta carpeta vienen diferentes archivos.

Como se puede ver en la siguiente ilustración vienen los archivos correspondientes al proyecto app (es como se ha llamado al proyecto) se va a hacer una breve explicación de cada uno de estos archivos.

- **app.html:** este html funciona como el html global en el cual dentro se importa las diferentes vistas de los componentes que se creen para la aplicación.
- **app.css:** es el css que se le impondrá a la aplicación de forma global.
- **app.spec.ts:** autogenerado por angular.
- **app.ts:** es el archivo interno, es un archivo typescript, parecido a javascript, hace referencia al html del mismo componente.
- **app.config.ts:** archivo de configuración donde se importarán librerías y elementos externos a nuestra aplicación.
- **app.routes.ts:** este archivo es muy importante ya que se definirán las rutas que utilizarán los componentes en la aplicación.

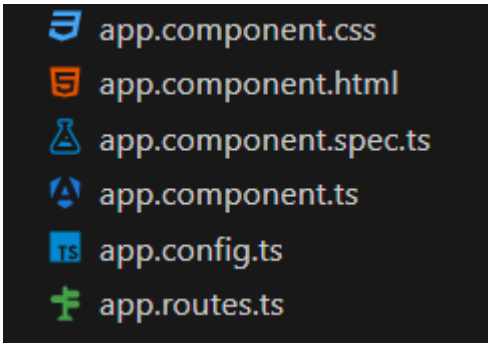

ILUSTRACIÓN 32. EJEMPLO CREACIÓN PROYECTO ANGULAR

Cada vez que se cree un componente en Angular se crearan los archivos css, html, spec.ts y ts, como se ha visto anteriormente haciendo referencia a los archivos de su componente y se trabajará importando estos componentes creados al componente proyecto llamado app.

Para concretar un poco más a fondo se ha de mencionar que se ha utilizado la última versión, Angular 17, el cual se hacen importaciones de componentes con el formato standalone. Ahora se va a proceder a explicar cómo anteriormente se ha hecho con ASP .NET Core la arquitectura, los layers, ciclo de vida, estructuración del proyecto y pruebas.

5.3.3.1 Arquitectura de Angular

Angular utiliza una arquitectura llamada “Clean Arquitecture” la cual su función es centralizar su solución en el dominio, esto significa, centrarse en su capa de negocio y no quedarse en un concepto técnico. Para ello se han utilizado las siguientes partes:

- **Entidades:** Objetos de la aplicación, esta capa es como la definición de las tablas en ASP .NET Core, tiene que haber una semejanza entre los modelos usados en el Back-End y las entidades del Front-End. Ningún cambio en la navegación de la aplicación ni utilización de esta debería someter un cambio a esta capa.
- **Casos de uso:** Esta capa contiene reglas comerciales y lógica de la aplicación, estos organizan el flujo de datos desde las entidades y hacia estas mismas. Se espera que los cambios en el funcionamiento de la aplicación generen cambios en la esta capa.

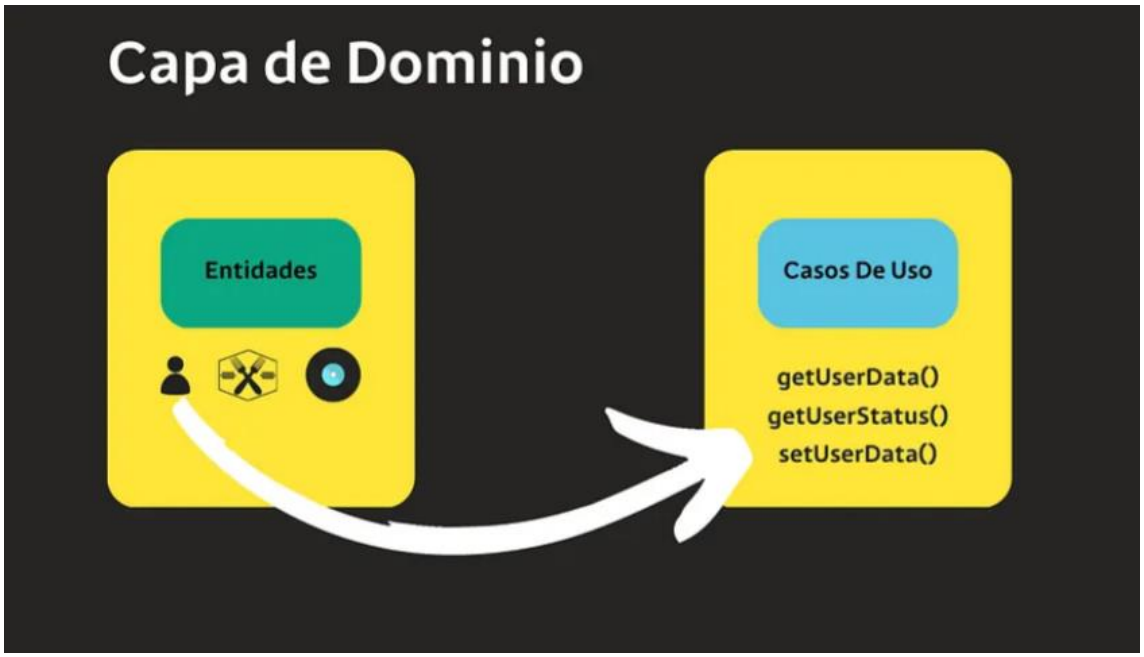

ILUSTRACIÓN 33. CAPA DOMINIO ANGULAR

- **Capa de infraestructura:** Contiene los siguientes elementos:
  - **Driven adapters:** son los adaptadores que permiten tener una conexión con el exterior.
  - **Entry points:** es la capa en la que se exponen los servicios, es común encontrarla en soluciones que conecten con Back-End.
  - **Helpers:** capa dedicada a ayudar a las demás con transformaciones de datos, operaciones o funciones útiles.
- **Capa de presentación:** implementa todo lo relacionado con elementos visuales con los que el usuario trabajará.

En el siguiente apartado se va a explicar los layers que se ha usado en el proyecto.

5.3.3.2 Layers en Angular

A lo largo del proyecto se han utilizado diferentes layers, se va a explicar uno por uno cuales han sido los esenciales: Modelos, Components y Services.

- **Modelos:** esta capa genera todas las entidades necesarias para el uso del proyecto, como se ha mencionado anteriormente debe de haber una correspondencia entre los modelos del Front-End y el Back-End.
- **Components:** donde vendrá la mayor parte de la lógica de nuestra aplicación, se crean los componentes que compondrán las vistas y el funcionamiento del proyecto. Cabe destacar que dentro de estos componentes se ha utilizado un Guard, es un comprobador para que gente no logeada o que esté sin permisos no pueda acceder a la aplicación.
- **Services:** esta capa es la encargada de hacer todas las llamadas a la API que se utiliza en el Back-end, transmitir y recibir los datos. Como se ha mencionado anteriormente en la memoria, estos datos se transmiten en formato JSON.

Posteriormente se muestra la ilustración de como resultaría la utilización de estas capas dentro de la carpeta app.

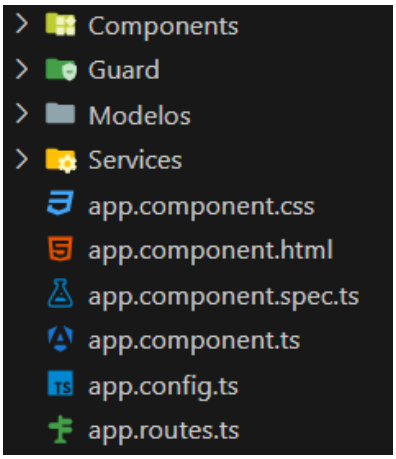

ILUSTRACIÓN 34. LAYERS ANGULAR

5.3.3.3 Ciclo de vida

Angular tiene un ciclo de vida complejo así que se explicará las 3 partes más usadas en este proyecto en el ciclo de vida: constructor, ngOnInit y ngAfterViewInit.

- constructor: en esta parte se definen los servicios y componentes externos necesarios para el buen uso del propio componente.
- ngOnInit: se ejecuta una vez el componente se ha inicializado, este es el evento utilizado para generar los datos iniciales de los componentes.
- ngAfterViewInit: se ejecuta cuando el componente se ha incializado al completo con sus vistas primarias y secundarias.

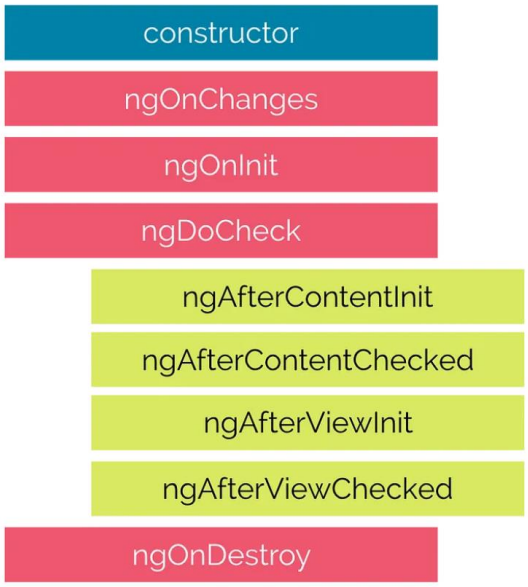

ILUSTRACIÓN 35. CICLO DE VIDA ANGULAR

Como se puede observar en la imagen anterior hay más partes, pero dado que no se han utilizado en el proyecto no van a ser explicadas.

5.3.3.4 Estructuración del proyecto

La estructuración del proyecto es clara dentro de la carpeta app, es donde se ha generado todos los componentes necesarios de la aplicación y utilizados en el componente app que es el global de la aplicación. El resto de archivos son autogenerados y completados por el propio Angular así que no se han explicado ni centrado en estos, como se puede ver en la siguiente ilustración.

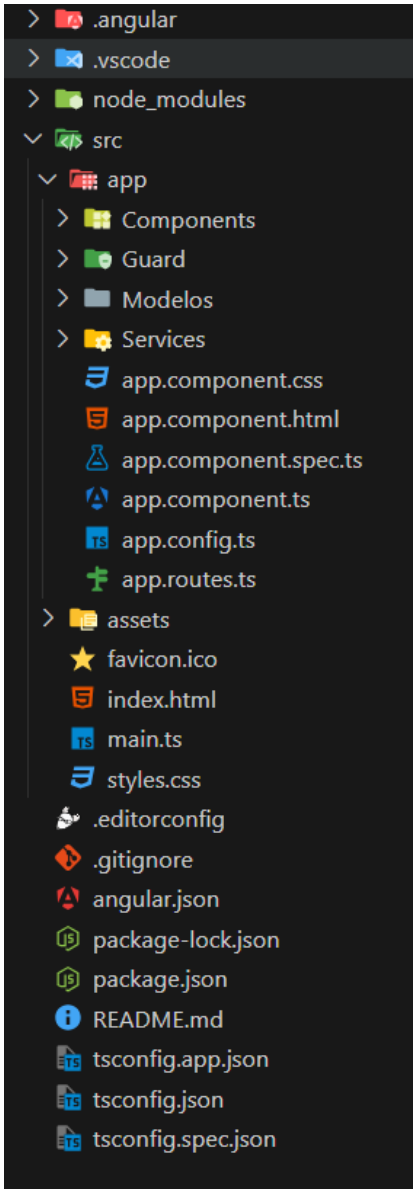

ILUSTRACIÓN 36. ESTRUCTURA PROYECTO ANGULAR

Como se ha mencionado anteriormente, se trabaja mediante componentes importados unos a otros. El funcionamiento del componente es el siguiente:

El html del componente contiene la vista del usuario final, complementado por el css y el ts, del que saca los valores necesarios para ser llamados por nuestra vista. Si se hace alguna navegación o alteración en la vista está llama al componente typescript asociado y hace la acción correspondiente, en el caso de que se necesiten valores del Back-End, el archivo typesctipt llama al servicio necesario, este comunica con el Back-End y se transfieren los valores necesarios.

5.3.3.5 Pruebas

Se tiene el mismo caso que anteriormente en ASP .NET Core, debido a la magnitud del proyecto, no se ha podido implementar pruebas unitarias de ningún tipo, sin embargo, esto no significa que no esté probado el proyecto, ya que, se ha ido probando a lo largo de todo su desarrollo mediante varias personas en funcionalidad y riesgo.

Cabe destacar que, por el objetivo del proyecto, ha tenido que haber siempre pruebas humanas dado que una de los objetivos de este proyecto es, que el profesor Juan Antonio Andrés Lalueza, va a hacer uso de este para impartir clases en su asignatura.

6. Resultados experimentales.

En general, enfocándonos en el resultado práctico, en el que implicaba encadenar dos tecnologías nunca usadas, y usarlas a la vez, ha sido muy satisfactoria por los siguientes motivos:

- **Conexión:** en el principio del proyecto había demasiadas dudas, ya que, al ser tecnologías nunca usadas anteriormente, era probable que fueran poco compatibles o que la dificultad fuera demasiado alta para su uso, sin embargo, una vez terminado el proyecto se puede decir que, aunque ha habido momentos de dificultad, en general se ha podido trabajar muy bien y se han conectado las dos tecnologías de una manera fluida y precisa.
- **Requisitos:** se han completado todos los requisitos que se han mencionado en la memoria por lo que, ha sido un proyecto satisfactorio con buenas sensaciones.
- **Aprendizaje:** las tecnologías usadas en nuestra aplicación son tecnologías consideradas de las más importantes hoy en día por lo que su aprendizaje además de haber sido muy divertido ha sido una parte importante en el proyecto.

Por último, se va a mostrar en las siguientes ilustraciones como han quedado las vistas finales de nuestra aplicación Banco VVBA.

Ventana de login

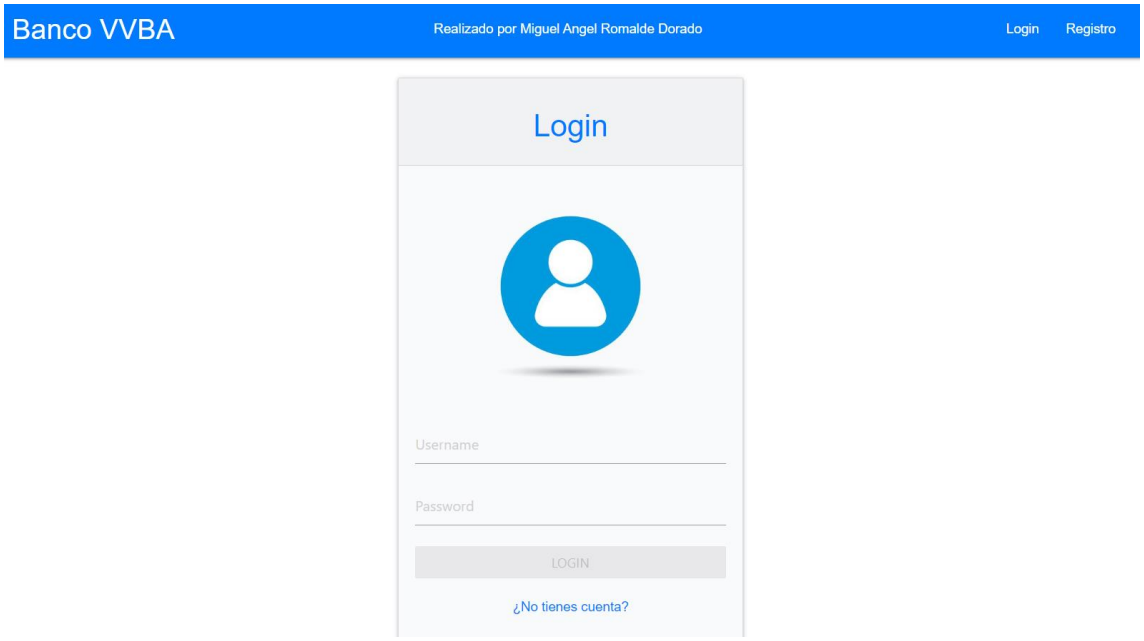

ILUSTRACIÓN 37.LOGIN FINAL

Registro

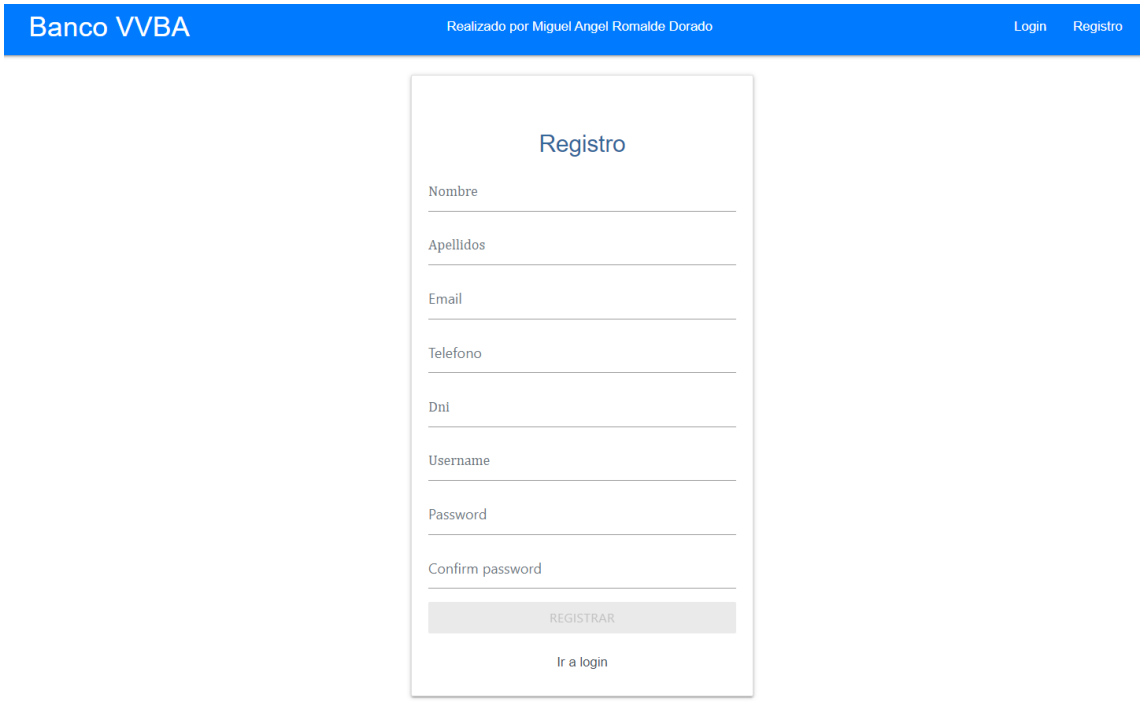

ILUSTRACIÓN 38. REGISTRO FINAL

Usuarios

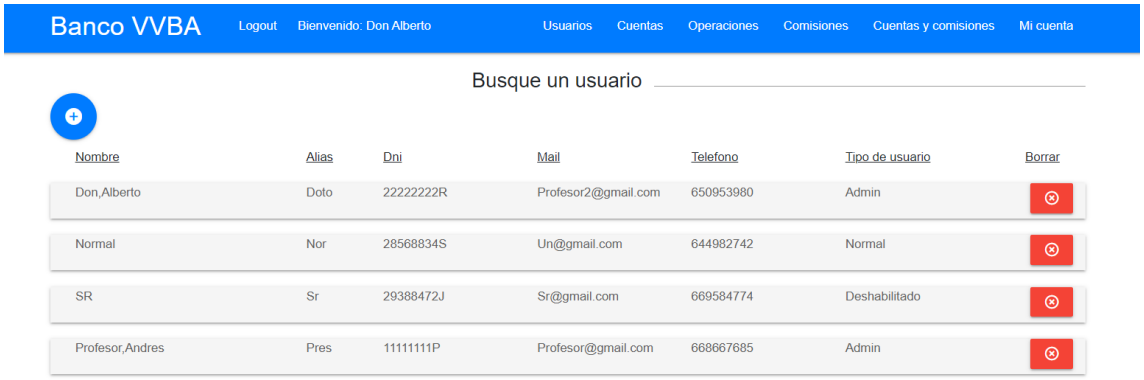

ILUSTRACIÓN 39. USUARIOS FINAL

Creación/editar del usuario por parte del administrador

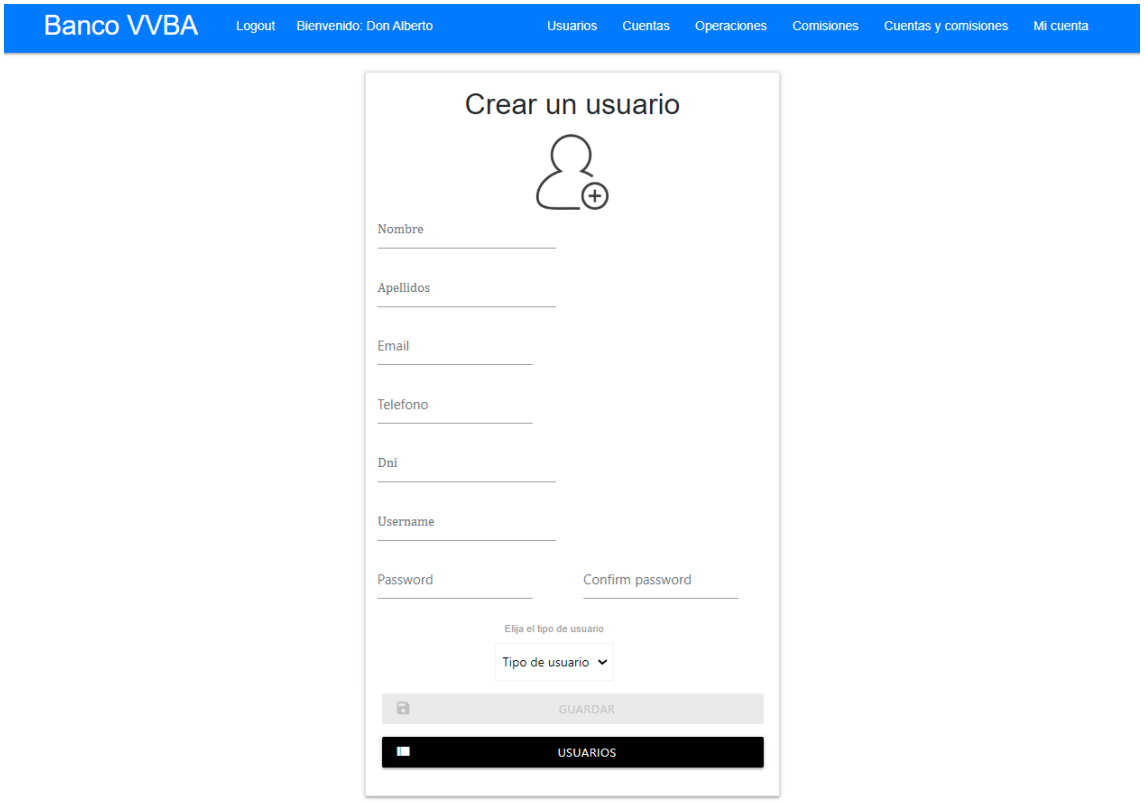

ILUSTRACIÓN 40. CREACION USUARIO

Cuentas

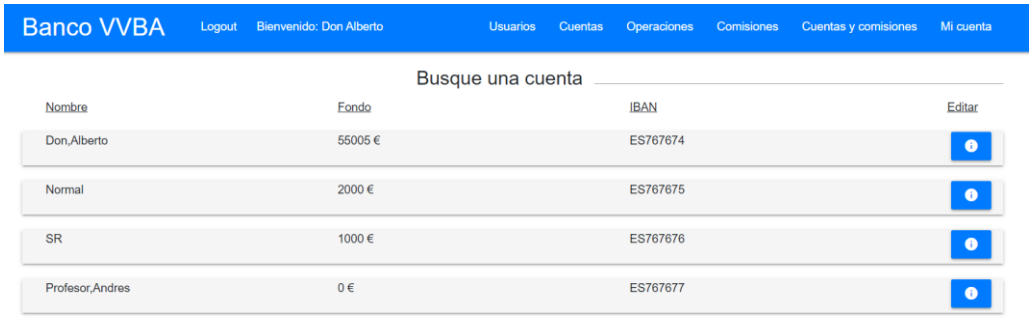

ILUSTRACIÓN 41. CUENTAS FINAL

Editar cuenta

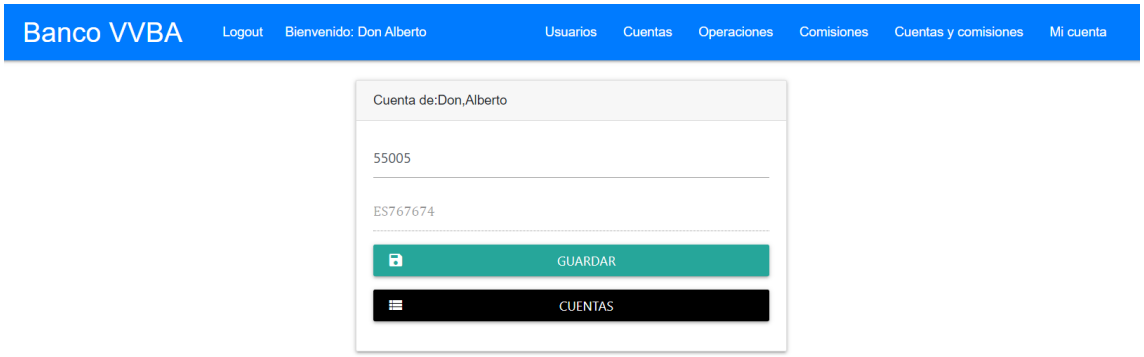

ILUSTRACIÓN 42. EDITAR CUENTA

Operaciones

ILUSTRACIÓN 43. OPERACIONES FINAL

Editar/crear operaciones

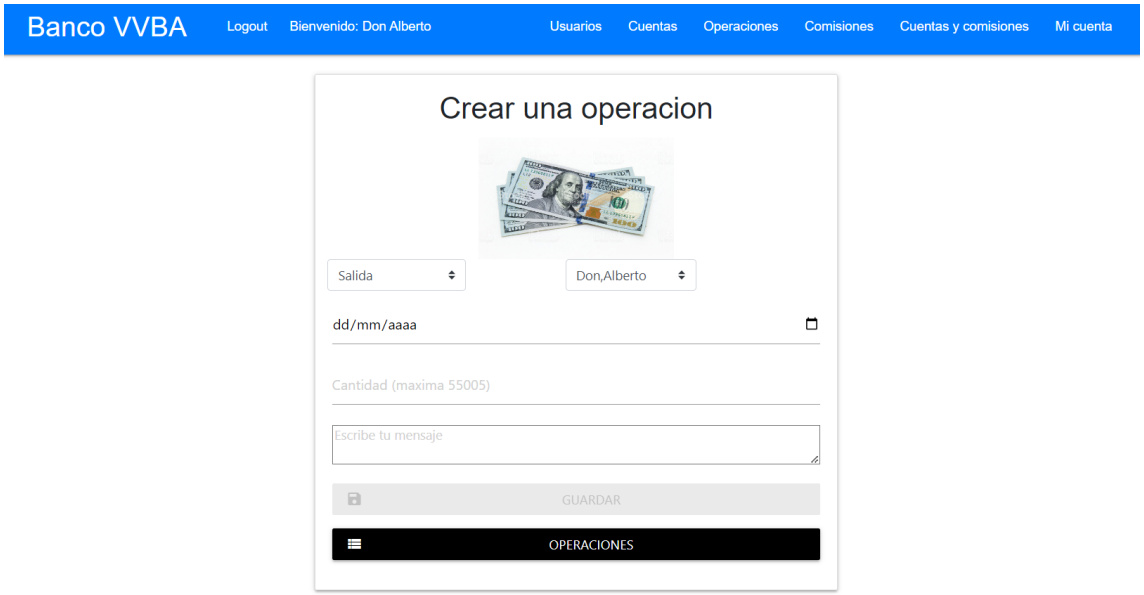

ILUSTRACIÓN 44. EDITAR OPERACIONES FINAL

Comisiones

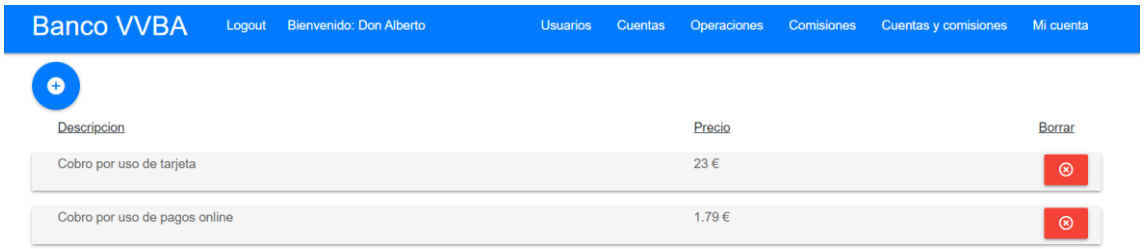

ILUSTRACIÓN 45. COMISIONES FINAL

Crear/editar comisiones

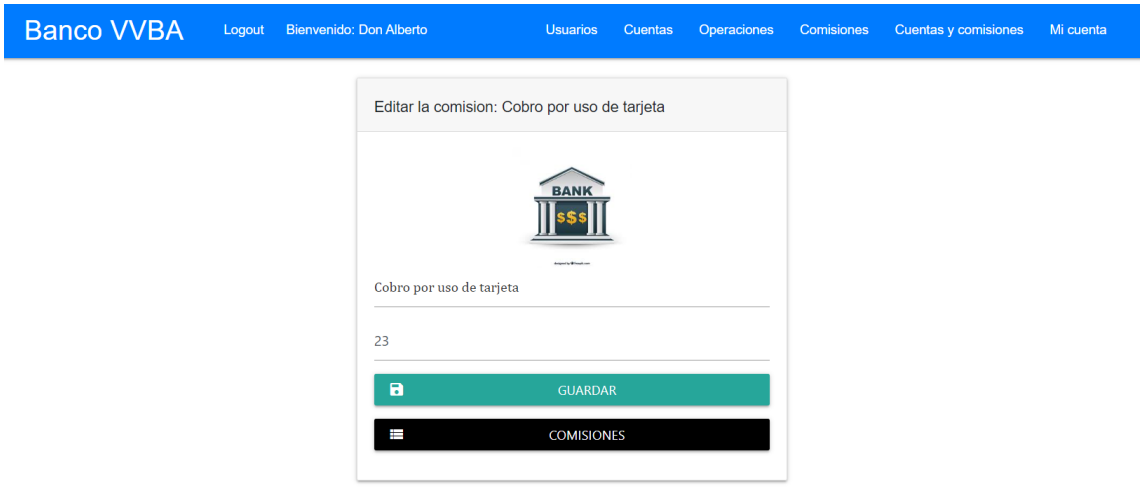

ILUSTRACIÓN 46. EDITAR COMISIONES

Cuentas comisiones

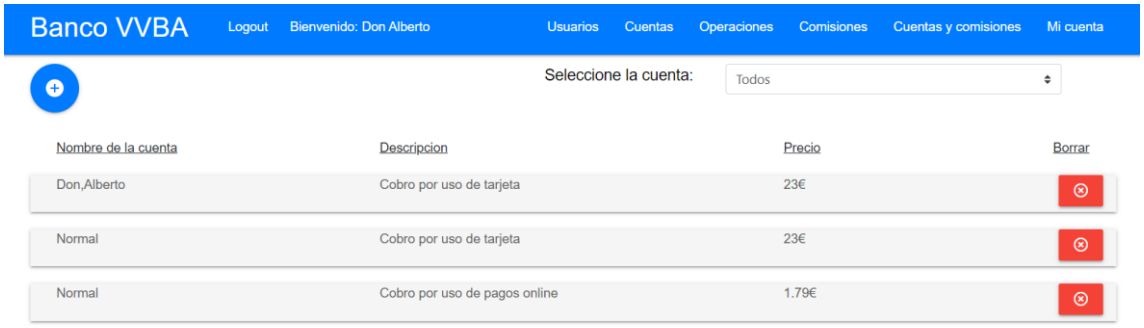

ILUSTRACIÓN 47. CUENTAS COMISIONES FINAL

Crear/editar cuentas comisiones

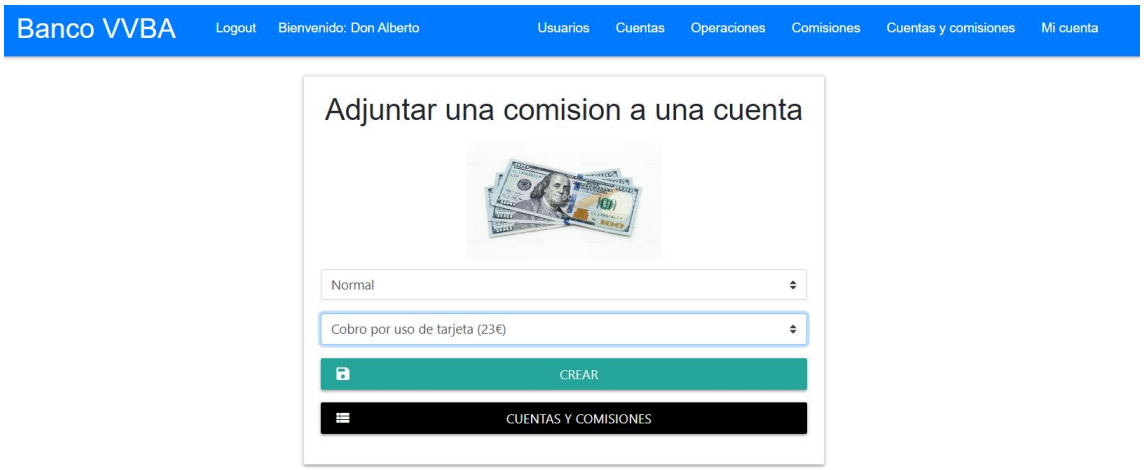

ILUSTRACIÓN 48. EDITAR CUENTAS COMISIONES

Mi cuenta

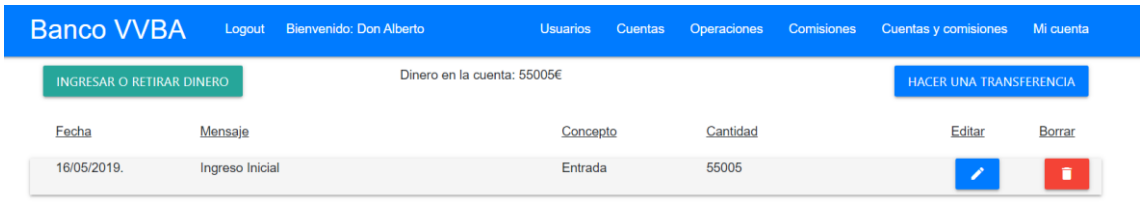

ILUSTRACIÓN 49. MI CUENTA FINAL

Ingresar/retirar dinero en mi cuenta

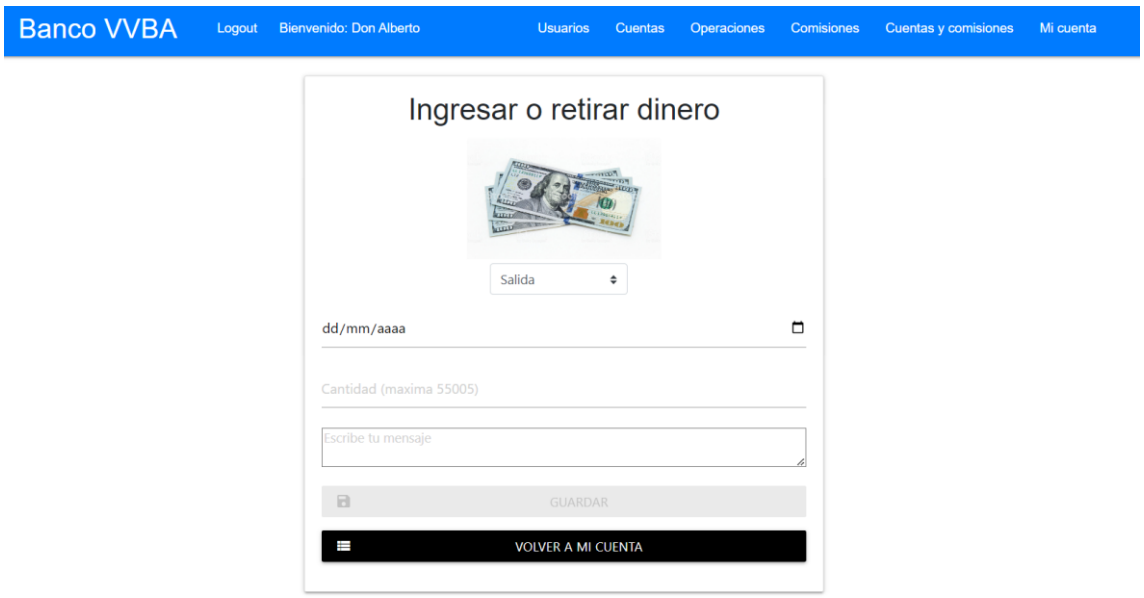

ILUSTRACIÓN 50. INGRESAR O RETIRAR DINERO FINAL

Transferencia

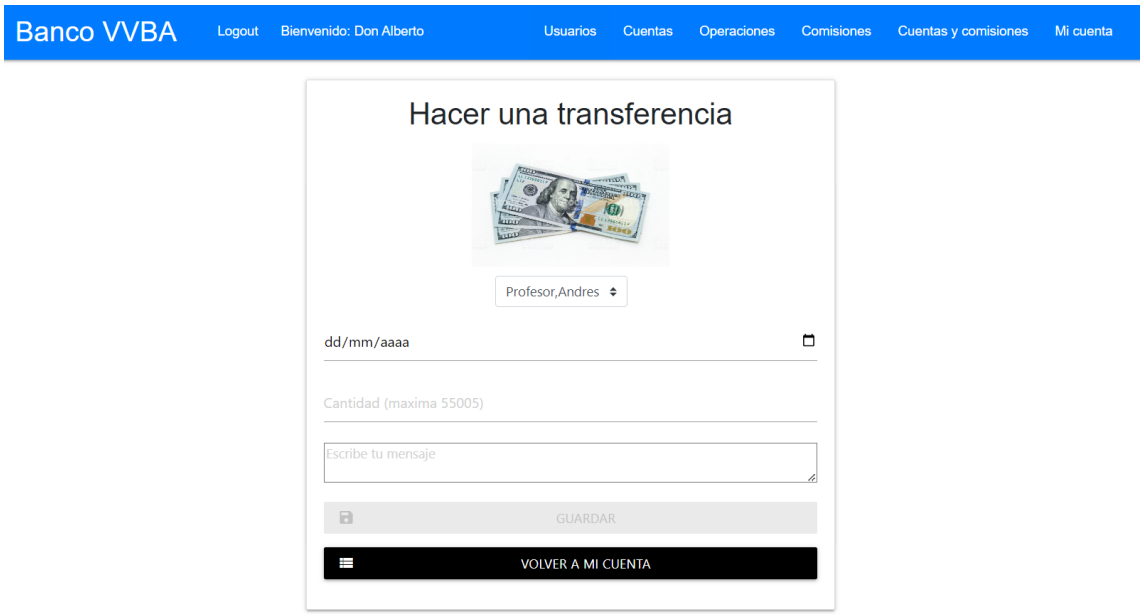

ILUSTRACIÓN 51. TRANSFERENCIAS FINAL

7. Conclusiones y trabajo futuro

Para comenzar, se ha cumplido el objetivo propuesto por el proyecto, el cual era realizar un estudio de las tecnologías más usadas en las empresas hoy en día y hacer una aplicación utilizando estas tecnologías. Se ha logrado realizando una aplicación bancaria el cual servirá como mockup para personas que no están socializadas con las aplicaciones bancarias que se utilizan hoy en día y puedan practicas a gusto y antojo. Además, la aplicación será utilizada en el ámbito de la enseñanza por el profesor ya mencionado anteriormente Juan Antonio Andrés Lalueza, el cual ha quedado satisfecho y agradecido por el trabajo realizado y el correcto funcionamiento que ha alcanzado la aplicación VVBA.

Gracias a la buena estimación ha sido un proyecto considerado exitoso, ya que, al haberse hecho un estudio previo de los requisitos y todo lo necesario para la aplicación se han podido cumplir bien los tiempos.

En cuanto a la línea de trabajo futuro, se podrían considerar las siguientes posibilidades:

- Implementar Jobs en la base de datos los cuales pudieran tener un “cobro” mensual de las comisiones en el salario de tu cuenta. Dependiendo del uso que se le haya podido dar a la aplicación.
- Se podría implementar un buscador el cual un usuario tenga la posibilidad de escribir lo que necesite hacer y la aplicación, por inteligencia artificial, haga una navegación hacia la página necesaria para hacer lo que el usuario ha pedido.
- Mejoras en seguridad, por ejemplo, se utilizaría autenticación biométrica para acceder a la aplicación, o incluso utilizar una autenticación con la cuenta de Google.
- Integración con pagos móviles, con ella se podría hacer pagos a través de códigos QR.
- Análisis de gastos, se podría crear una herramienta que te analice los gastos del último mes y haga una muestra de informes visuales.
- Soporte multilingüe, en la que se añadiría a la aplicación la posibilidad de cambiar de idioma para que más gente pudiera usarla.

8. Bibliografía

[1] Angular Team. (n.d.). *Creación de proyecto Angular y Tour of Héroes*. Angular. Retrieved September 24, 2024, from https://v17.angular.io/tutorial/tour-of-heroes

[2] Todo Nube. (2021, May 17). *Tutorial de creación de proyecto ASP .NET Core con Entity Framework Core* [Video]. YouTube. Retrieved September 24, 2024, from https://www.youtube.com/watch?v=fqR-WbmX8PM&t=2025s

[3] Todo Nube. (2021, April 12). *Conectividad con ASP .NET Core* [Video]. YouTube. Retrieved September 24, 2024, from https://www.youtube.com/watch?v=OZGdKYzUYvU&list=PLjC4UKOOcfDQtElvsn1ZCAHatLtqDrTgQ

[4] Microsoft. (n.d.). *Uso de paquetes NuGet*. Microsoft Learn. Retrieved September 24, 2024, from https://learn.microsoft.com/en-us/nuget/consume-packages/install-use-packages-powershell

[5] Stack Overflow. (n.d.). *Importación de componentes Standalone*. Stack Overflow. Retrieved September 24, 2024, from https://stackoverflow.com/questions/78035343/toastr-implementation-in-angular17-at-standalone-components

[6] Angular Team. (n.d.). *Conexiones HTTP con el Back-End desde Angular*. Angular. Retrieved September 24, 2024, from https://angular.dev/guide/http/setup

[7] Mik3Lov3st3ch. (2021, July 12). *Mapear modelos en Angular*. Medium. Retrieved September 24, 2024, from https://medium.com/@mik3lov3st3ch/mapping-api-responses-to-models-in-angular-using-a-model-adapter-38a82b348fe5

[8] Programming with Mosh. (2020, February 5). *Tutorial de uso de Guard* [Video]. YouTube. Retrieved September 24, 2024, from https://www.youtube.com/watch?v=dsVKoSyQZ98

[9] Microsoft. (n.d.). *ASP.NET Core documentation*. Microsoft Docs. Retrieved September 24, 2024, from https://docs.microsoft.com/aspnet/core/?view=aspnetcore-7.0

[10] MacDonald, A. (2019). *ASP.NET Core 3.0: The complete guide to building web applications with .NET Core 3.0*. Packt Publishing.

[11] Kranthi, K. (2020). Implementing security in ASP.NET Core applications. *Code Project*. Retrieved September 24, 2024, from https://www.codeproject.com/Articles/5256377/Implementing-Security-in-ASP-NET-Core-App

[12] Angular. (n.d.). *Angular documentation*. Angular.io. Retrieved September 24, 2024, from https://angular.io/docs

[13] Tslim, G. (2021). *Angular 11 by Example: Build real-world applications with Angular 11 and TypeScript*. Packt Publishing.

[14] Vasan, S. (2021). 10 best practices for Angular developers. *Medium*. Retrieved September 24, 2024, from https://medium.com/swlh/10-best-practices-for-angular-developers-87f6875e3c92

[15] Node.js Foundation. (2023). *Node.js - JavaScript runtime*. Node.js Documentation. https://nodejs.org/en/docs.

[16] Django Project. (2023). *Framework web de alto nivel para desarrollo rápido en Python*. Django Documentation. https://docs.djangoproject.com/.

[17] Ruby on Rails Team. (2023). *Ruby on Rails: A web-application framework that includes everything needed to create database-backed web applications*. Rails Guides. https://guides.rubyonrails.org/.

[18] Pivotal Software, Inc. (2023). *Spring Boot: Framework para aplicaciones Java*. Spring Boot Documentation. https://spring.io/projects/spring-boot.

[19] Flask Documentation. (2023). *Flask: Microframework para desarrollo web en Python*. Flask Documentation. https://flask.palletsprojects.com/en/latest/.

[20] Microsoft Learn. (2023). *ASP.NET Core: Framework para construir aplicaciones web modernas en .NET*. Microsoft. https://learn.microsoft.com/en-us/aspnet/core.

[21] Express.js Team. (2023). *Express.js: Minimalist web framework for Node.js*. Express.js Documentation https://expressjs.com/.

[22] Ashkenas, J. (2010). Backbone.js: Lightweight JavaScript Library. https://backbonejs.org

[23] Bynens, M. (2020). Preact Documentation. Preact.js. https://preactjs.com

[24] Cooper, A. (2023). Building Web Applications with React and Preact. Addison-Wesley. Eames, S. (2018). Introduction to Svelte. https://svelte.dev

[25] Flanagan, D. (2020). JavaScript: The Definitive Guide (7ª ed.). O'Reilly Media.

[26] Freeman, E., Robson, E., Bates, B., & Sierra, K. (2020). Head First Design Patterns (2ª ed.). O'Reilly Media.

[27] Hoch, M. (2021). Understanding Vue.js Framework. Packt Publishing.

[28] Igou, C. (2022). Mastering Angular (2ª ed.). Packt Publishing.

[29] Lecheta, J. (2022). Developing with React: Components and Virtual DOM. Apress.

[30] Sabin-Wilson, L. (2021). JavaScript Frameworks for Modern Web Development. Wiley.