Banco VVBA



**Realizado por:**

MIGUEL ANGEL ROMALDE DORADO

**Dirigido por:**

JUAN ANTONIO ORTEGA RAMIREZ

Tabla de contenidos

1. INTRODUCCIÓN

Este trabajo pretende analizar y explorar alternativas disponibles en el mercado tecnológico ayudando y facilitando a la población, a entender cómo funciona una aplicación de banco sin ningún riesgo desde la comodidad de su hogar.

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 el miedo a la utilización de aplicaciones del mismo estilo.

Cabe destacar que se usarán tecnologías actuales el cual ayuden al propietario de la aplicación a su entrada a diferentes empresas. Posteriormente se hará un estudio de dichas tecnologías y se hará una selección acorde a los requerimientos.

* 1. ABSTRACT

This work aims to analyse and explore alternatives available on the technology market, helping and facilitating the population to understand how a banking application works without any risk from the comfort of their home.

On the other hand, to complete the study case and objective of the work in its practical form, the aim is to develop an application for the Windows desktop platform, collaterally, with the application it is intended to promote the independence of users when working with a banking application and reduce the fear of using applications of the same style.

It should be noted that current technologies will be used which will help the owner of the application to enter different companies. Subsequently, a study of these technologies will be carried out and a selection will be made according to the requirements.

* 1. AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mi tutor, Juan Antonio Ortega Ramirez, 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.

Por último, gracias a mi familia y amigos que han estado apoyándome en momentos donde creía que no iba a conseguir finalizarlo a tiempo y me daban ganas de rendirme, es gracias ellos que este proyecto ha salido adelante.

1. ESTUDIO DEL ARTE

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

* 1. Tecnologías de FrontEnd
     1. React

 **Descripción:** Biblioteca de JavaScript para construir interfaces de usuario.

 **Características:**

* Componentes reutilizables.
* Virtual DOM para mejorar el rendimiento.
* Amplio ecosistema de herramientas y bibliotecas.

 **Ventajas:** Flexibilidad, gran comunidad, buen rendimiento.

 **Desventajas:** Puede ser complejo de aprender para principiantes debido a su ecosistema.

* + 1. Angular

 **Descripción:** Framework de aplicaciones web basado en TypeScript.

 **Características:**

* Estructura MVC.
* Inyección de dependencias.
* Herramientas integradas como Angular CLI.

 **Ventajas:** Estructurado, soporte para aplicaciones grandes, dos vías de enlace de datos.

 **Desventajas:** Curva de aprendizaje empinada, complejo.

* + 1. Vue.js

 **Descripción:** Framework progresivo de JavaScript para construir interfaces de usuario.

 **Características:**

* Integración gradual.
* Reactividad.
* Componentes.

 **Ventajas:** Fácil de aprender, flexible, buena documentación.

 **Desventajas:** Menos recursos y herramientas en comparación con React y Angular.

* + 1. Svelte

 **Descripción:** Framework para construir interfaces de usuario que compila eficientemente.

 **Características:**

* Compilación en tiempo de desarrollo.
* Sin virtual DOM.
* Código más simple.

 **Ventajas:** Rendimiento muy alto, menos código.

 **Desventajas:** Comunidad más pequeña, no está tan desarrollada como otras opciones.

* + 1. Ember.js

 **Descripción:** Framework de JavaScript para construir aplicaciones web ambiciosas, similar a Vue.js pero con más dificultad y complejidad.

 **Características:**

* Convención sobre configuración.
* Herramientas de desarrollo integradas.
* Enrutamiento robusto.

 **Ventajas:** Productividad alta, estructura sólida.

 **Desventajas:** Curva de aprendizaje empinada, menos flexible.

* + 1. Backbone.js

 **Descripción:** Biblioteca de JavaScript para estructurar aplicaciones web con modelos y vistas.

 **Características:**

* Ligero.
* MVP (Model-View-Presenter).
* Sincronización con servidor RESTful.

 **Ventajas:** Ligero, flexible.

 **Desventajas:** Requiere más configuración, menos soporte oficial.

* + 1. Preact

 **Descripción:** Alternativa ligera a React.

 **Características:**

* API similar a React.
* Tamaño muy pequeño y eficiente.
* Virtual DOM.

 **Ventajas:** Muy ligero y un gran rendimiento.

 **Desventajas:** Menos características integradas, menos comunidad y menos información.

* 1. Tecnologías del BackEnd
     1. Node.js

 **Descripción:** Entorno de ejecución para JavaScript del lado del servidor.

 **Características:**

* Event-driven.
* Non-blocking I/O.
* Gran ecosistema (NPM).

 **Ventajas:** Trabajo de forma asíncrona, gran comunidad.

 **Desventajas:** No adecuado para tareas intensivas en CPU.

* + 1. Django

 **Descripción:** Framework web de alto nivel en Python.

 **Características:**

* Estructura MVC.
* ORM incorporado.
* Herramientas de administración.

 **Ventajas:** Seguridad, desarrollo rápido y API REST integrado.

 **Desventajas:** Pesado, menos flexible.

* + 1. Ruby on Rails

 **Descripción:** Framework web en Ruby.

 **Características:**

* Convención sobre configuración.
* Herramientas integradas.
* ORM (ActiveRecord).

 **Ventajas:** Productividad, orientado a prototipos rápidos.

 **Desventajas:** Rendimiento, curva de aprendizaje elevada.

* + 1. Spring boot

 **Descripción:** Framework para aplicaciones Java.

 **Características:**

* Microservicios.
* Configuración automática.
* Seguridad incorporada.

 **Ventajas:** Potente, escalable.

 **Desventajas:** Configuración compleja, pesado.

* + 1. Express.js

 **Descripción:** Similar a Node.js, minimalista y ligero.

 **Características:**

* Middleware.
* Ruteo sencillo.
* Gran comunidad.

 **Ventajas:** Ligero, flexible.

 **Desventajas:** Requiere más configuración, menos estructurado.

* + 1. Flask

 **Descripción:** Microframework web en Python, similar a Django pero menos complejo.

 **Características:**

* Minimalista.
* Extensible.
* WSGI.

 **Ventajas:** Ligero, fácil de aprender.

 **Desventajas:** No tiene muchas herramientas integradas.

* + 1. ASP.NET Core

 **Descripción:** Framework para construir aplicaciones web modernas en .NET.

 **Características:**

* Cross-platform.
* Inyección de dependencias.
* Buen rendimiento.

 **Ventajas:** Integración con el ecosistema .NET además un buen rendimiento.

 **Desventajas:** Curva de aprendizaje y más complejo para principiantes.

* 1. Selección de tecnologías

Las tecnologías de FrontEnd y BackEnd 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.

* + 1. FrontEnd

En nuestro caso se ha elegido utilizar Angular en el apartado de FrontEnd 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.

1. **Estructura Modular**

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

1. **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.

1. **Inyección de dependencias**

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

1. **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.

1. **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.

* + 1. BackEnd

Para la parte del BackEnd se ha decidido utilizar ASP.NET Core 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.

1. **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 en nuestro caso, no es relevante dado que vamos a hacer la aplicación solo para Windows.

1. **Inyección de dependencias integrada**

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

1. **Modularidad y flexibilidad**

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

1. **Compatibilidad con Microservicios**

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

1. **Seguridad**

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

1. **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, otras de las cosas más importantes es que para hacer este TFG, era necesario usar tecnologías que ayudaran personalmente a la incorporación al mundo laboral. Son tecnologías de las más usadas y nuevas, ambas tienen un gran rendimiento y un soporte a largo plazo, por lo que son ideales para la realización de este proyecto.

1. 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.

* 1. Elicitación de requisitos del producto
     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 1. REQUISITOS DE INFORMACIÓN

* + 1. Requisitos funcionales

A continuación, veremos 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 2. Requisitos funcionales

* + 1. Requisitos no funcionales
       1. Requisitos de calidad

En este apartado se verán 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 3. Requisitos de calidad

* + - 1. Requisitos de implementación

En este apartado se expondrán los requisitos de implementación.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Descripción | Prioridad | Criterio de aceptación |
| RIM-01 | Se implementara 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 4. Requisitos de implementación

* + - 1. Requisitos de seguridad

En este apartado veremos 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-05 | Los datos sensibles han de estar encriptados | Alta | Datos como contraseñas estarán encriptados con un algoritmo fiable |
| 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 |

* 1. 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

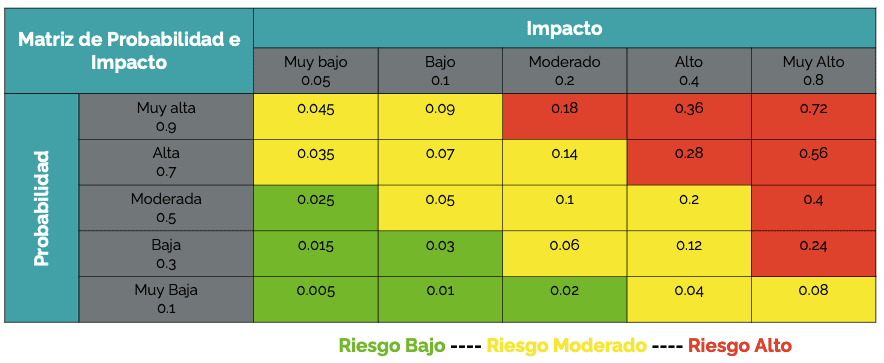


Ilustración 1. Matriz de probabilidad e impacto

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 5. Tabla de riesgos

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

* 1. Viabilidad

En este caso hemos hecho una encuesta anónima en la cual hay 6 preguntas hechas mediante la plataforma de google forms, aunque también hay una persona que es profesor de economía que pretende usar la aplicación en sus clases.

1. VERSIÓN DE COSTES