Informe Final. Clash Royale
Ingeniería de Software + Base de Datos.

Karen Dianelis Cantero Lopez. C311
Luis Alejandro Rodriguez Otero C311
Hector Miguel Rodriguez Sosa C311
Sebastian Suarez Gomez C311

1 Introducción

El producto tiene como objetivo analizar los datos de los usuarios de Clash Royale y proporcionar la información más relevante sobre su comportamiento, patrones de juego, estrategias y rendimiento. Se utilizan análisis de métricas de participación y evaluación de la experiencia del usuario. En el presente informe se describe el proyecto. Se presentan los requerimientos específicos funcionales, no funcionales y del entorno. Las funcionalidades del producto se muestran a partir de un modelo de casos. En la arquitectura del proyecto se presentan la arquitectura general y la arquitectura de diseño del backend. En el modelo de datos se muestra el modelo de base de datos. Estos y otros aspectos significativos para la realización del producto se abordan a continuación.

2 Requerimientos Específicos

Requisitos Funcionales

1. El usuario puede registrarse e iniciar sesión.
2. El usuario puede visualizar gráficamente la popularidad del juego; es decir, puede ver gráficamente la cantidad de batallas y desafíos en el año.
3. Realizar batallas entre jugadores simulando el resultado. Visualizar las cartas con sus características.
4. Realizar operaciones CRUD sobre las entidades.
5. Realizar consultas a la base de datos y exportar los resultados de dichas consultas en archivos .pdf o .csv.
6. Realizar un análisis del mazo del jugador.
7. Realizar análisis de métricas de participación: el sistema es capaz de analizar y proporcionar información detallada sobre la participación de los usuarios en el juego.

Requisitos No Funcionales

1. Seguridad: El sistema garantiza la confidencialidad, integridad y autenticación de los datos de los usuarios. El acceso a la plataforma es controlado por nombre de usuario y contraseña. Una vez el usuario esté autenticado en la plataforma, se le asignará un token jwt. En caso de este expirar luego de haberse iniciado sesión, se necesitará que se autentique de nuevo.
2. Rendimiento: El sistema es eficiente y tiene un rendimiento óptimo. Asegura tiempos de respuesta rápidos y capacidad para manejar el volumen de datos y usuarios sin degradar la calidad del servicio.
3. Usabilidad: La interfaz del sistema es intuitiva, fácil de usar. Se siguen las mejores prácticas de diseño de experiencia de usuario (UX) para garantizar una interacción fluida y positiva.

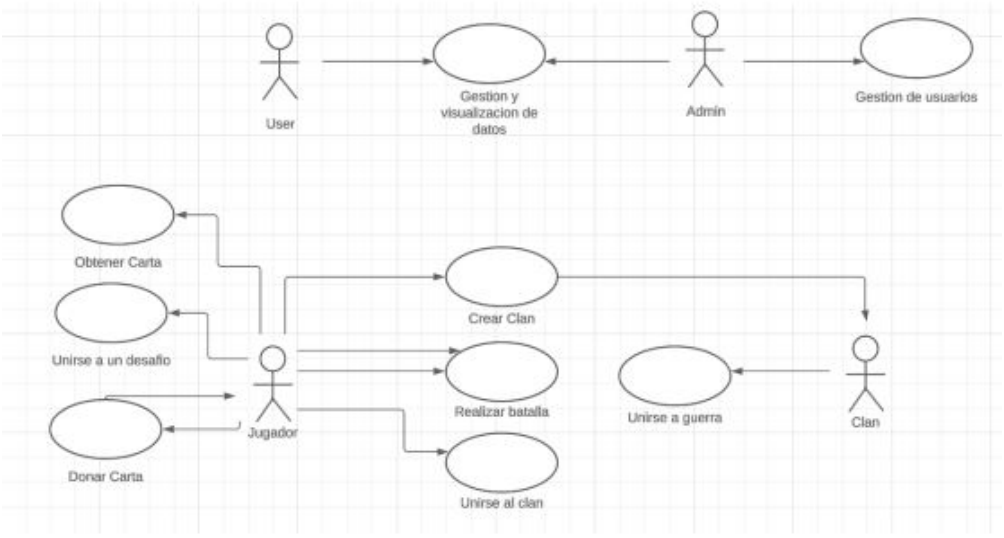
4. El sistema alacena todos los datos en una base de datos SQL Server. A través del sistema se pueden realizar consultas requeridas y algunas parametrizadas.
5. El sistema cuenta con un apartado FAQ donde se le darán respuesta a algunas preguntas técnicas que puedan tener los usuarios.
6. Disponibilidad: Existen tres tipos de usuarios (user, admin y superadmin), donde a cada uno se le garantiza el acceso a un determinado tipo de información.
7. Diseño e implementación: En el proyecto se emplea una arquitectura monolítica. Para la construcción del Backend, se usó .NET Core con C# y para el Frontend, se usó Angular. En el Backend, se usó una arquitectura de diseño basada en un dominio conocida como DDD. Para el trabajo con la base de datos se usó Entity Framework Core.

Requisitos de entorno

1. Framework: .NET Core 7 y Angular.
2. Lenguaje de programación: C# y Typescript.

3 Funcionalidades del producto

Modelo de casos



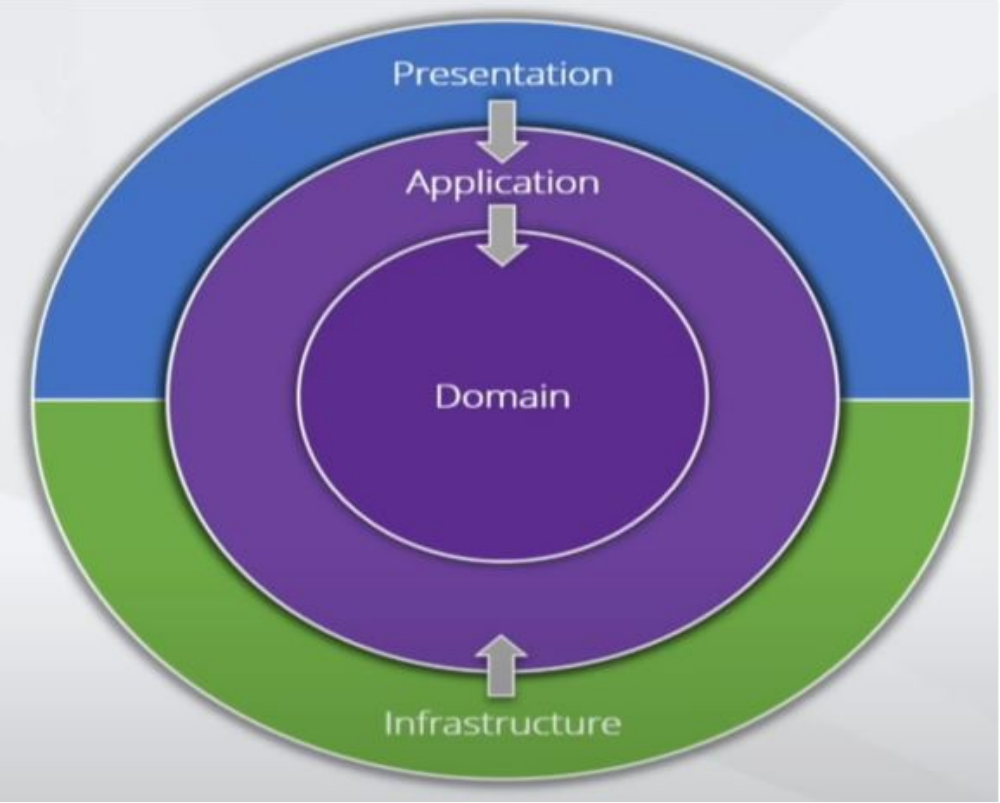
4 Enfoque Metodológico

Para la creación, se dividió el proyecto teniendo en cuenta las destrezas de los individuos y cada uno trabajó en un sector específico. El jefe de equipo repartió las tareas basándose en lo anterior.

5 Arquitectura

En el proyecto en general, se utiliza una arquitectura monolítica donde lo componentes se integran en una sola unidad y son compatibles entre sí.

Diseño de Backend



Se escogió esta arquitectura de diseño para tener el proyecto correctamente separado y garantizar las mejores prácticas de programación. Se emplea el patrón de diseño CQRS, donde se separan las responsabilidades en commands y queries.

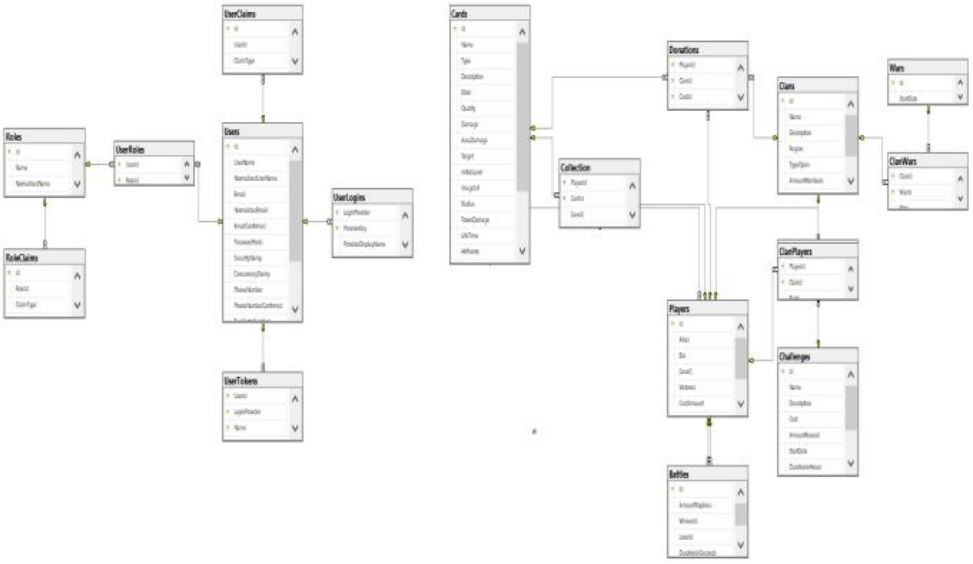
1. Domain: Entidades, excepciones, errores y posibles partes que no necesiten de ninguna otra capa.
2. Application: Interfaces y posibles objetos que interactúen con el Domain.
3. Infrastructure: Diferentes implementaciones de las interfaces de repositorio y todo lo relacionado con la base de datos.

6 Patrones de Visualización

Una gráfica de líneas muestra la popularidad del juego por mes (cantidad de batallas en el año). Mientras el proyecto avance se irán agregando más formas de visualización de datos.

7 Modelo de Datos

Base de datos:



8 Diccionario de Datos

9 Esquema de las clases definidas

10 Opciones del sistema

11 Salidas del sistema

12 Modificaciones sobre el diseño de base de datos propuesto al inicio