



Propuesta de Trabajo de Servicio Comunitario

Memoria Web

Propuesta elaborada por:
Arturo Yepez | 15-11551
Estudiante Ing. de la Computación



Introducción

Propósito del Documento

Este documento de Propuesta de Proyecto de Servicio Comunitario cubre el esquema del desarrollo del proyecto de trabajo para lograr los términos del acuerdo entre el estudiante Arturo Yepez y ANAR, y para impulsar la cooperación y minimizar la posibilidad de conflicto o confusión entre las organizaciones.

Además del alcance del trabajo, el calendario, los objetivos del proyecto, las normas y las definiciones de éxito, este documento también abordará otras informaciones diversas, como las consideraciones de seguridad, las restricciones de hardware y software, el apoyo posterior al proyecto, entre otras.

Este documento de especificación sirve de base para los desarrolladores, proporcionándoles los detalles necesarios para el desarrollo de características nuevas o existentes para el proceso de negocio seleccionado.

Objetivos del Producto Mínimo Viable (MVP)

Los objetivos y beneficios esperados luego de finalizado el desarrollo son al menos los siguientes:

- Terminar la aplicación web “Memoria Web” que permita a los usuarios realizar la actividad de recreación del juego en base a memorizaciones.
- El trabajo desarrollado va a consistir principalmente en hacer una separación de servicios de la aplicación para crear un servicio de Backend independiente, que posea características de uso multiplataforma
- El Backend debe tener la capacidad de ser expandido para poder utilizarse en un futuro con las otras aplicaciones de ANAR.
- Debe realizarse una migración del servicio actual de base de datos de “Memoria Web” a MySQL.
- El Backend debe estar hecho con NodeJS



Descripción y Necesidades

Descripción del Problema

La actual arquitectura de la aplicación “Memoria Web” no permite una fácil e intuitiva conexión de lo almacenado en su base de datos con los demás usuarios para crear juegos multijugador.

Además, la tecnología utilizada para la base de datos, crea un problema muy grande de persistencia de la información entre distintas sesiones de usuarios.

Solución Propuesta

Hacer un cambio de arquitectura para toda la aplicación, con bases para que en un futuro pueda ampliarse para que sea base interconectada de todos los proyectos de ANAR de una forma estandarizada por el medio.

La solución propuesta consiste en implementar un servicio de Backend implementado en un entorno NodeJS, con los suficientes módulos iniciales como para mantener las funcionalidades (principalmente de multijugador online y manejo de accesos) del juego “Memoria Web”.

Sin embargo, lo más importante de este proyecto consiste en la migración de la base de datos PouchDB a MySQL, es decir, de una base de datos no estandarizada No-SQL a una estandarizada SQL.

Actuales Inconvenientes

- Todos los cambios efectuados han de ser conectados/actualizados en el Frontend luego de que estén listos, lo que conlleva que se necesite invertir tiempo para actualizarse.

Producto

Propósitos del Sistema

La solución propuesta gira en torno a la construcción de un Backend, donde al final del desarrollo este debe cumplir con las siguientes características:

- **Autenticación:** Debe existir un módulo de autenticación, que permita registrar usuarios y su información básica.



- **Manejo de Usuarios:** El sistema debe permitir un manejo completo de usuarios, establecido por la arquitectura CRUD (*Create, Read, Update, Delete*)
- **Manejo del Juego:** Todas las estructuras asociadas al juego “Memoria Web” tales como cartas, niveles, resultados, etc; deben de poder ser controlables (con la misma arquitectura CRUD).
- **Almacenamiento de Datos:** El sistema debe hacer todas las operaciones relacionadas al almacenamiento de datos en la base de datos MySQL, mediante una ORM.
- **Autocontenido** (opcional): Para facilitar el desarrollo y los distintos ambientes de pruebas o dispositivos donde se pueda ejecutar, se desea instalar mediante contenedores de Docker.

Arquitectura

Metodología

Tener una metodología de capas separadas basada en responsabilidades (Modelos, Controladores, Servicios, Middlewares, etc.) la que facilitará las pruebas unitarias, la depuración y la ampliación del código en un futuro.

Paradigma

Multi-Paradigma: manejado por eventos, programación orientada a objetos, middleware.

Patrón de diseño: Dirigido por eventos

Node.js hace un uso extensivo de los eventos que es una de las razones detrás de su velocidad cuando comparada con otras tecnologías similares. Una vez que iniciamos un servidor Node.js, éste inicializa las variables y funciones y luego escucha la ocurrencia de un evento.

La programación dirigida por eventos se utiliza para sincronizar la ocurrencia de múltiples eventos y para hacer el programa lo más simple posible. Los componentes básicos de un programa dirigido por eventos son:

- Un **callback** (llamado manejador de eventos) es llamado cuando se dispara un evento.



- Un **bucle de eventos** que escucha los disparos de eventos y llama al manejador de eventos correspondiente para ese evento

Lenguaje de Programación

Javascript o Typescript en caso de ser permitido.

Librerías

Para la elaboración del Backend, utilizaremos las siguientes librerías

Nombre	Descripción
Nodejs	Node.js es un entorno de ejecución multiplataforma, construido sobre V8, un motor JavaScript de código abierto de alto rendimiento. Para garantizar un rendimiento excepcional.
Express	Express es un framework de aplicaciones web Node.js mínimo y flexible que proporciona un sólido conjunto de características para aplicaciones web y móviles.
Sequelize	Sequelize es un ORM de Node.js basado en promesas para Postgres, MySQL, MariaDB, SQLite y Microsoft SQL Server. Cuenta con un sólido soporte de transacciones, relaciones, carga ansiosa y perezosa, replicación de lectura y más.
Jest	Jest es una librería que ofrece un marco de pruebas de JavaScript con un enfoque en la simplicidad.

Versionamiento del Sistema

Especificación semántica del versionamiento a utilizar en el proyecto. Dado una versión MAYOR.MENOR.PARCHE, se puede identificar los cambios de la siguiente manera:

- MAYOR cuando se hacen cambios incompatibles entre versiones del API.
- MENOR cuando se agrega funcionalidad de una forma que sea compatible con versiones anteriores.
- PARCERAS cuando se introducen parches para problemas de una forma compatible con versiones anteriores.



Documentación del API

Se realizará en una colección de la herramienta Postman, que será actualizada con cada cambio en el mismo repositorio.

Lista de rutas iniciales del API

El sistema ha de representarse como un Backend, desde el cual se ha de poseer una API para que se pueda conectar a los diferentes servicios de Frontend que han de consumirlo. Una consecuencia de este hecho es que el sistema contará con una serie de endpoints según las distintas funcionalidades a implementar.

Actualmente, de forma inicial se propone agrupar los endpoints en una serie de módulos, desde la cual luego se van a consumir las distintas funcionalidades:

- */auth* - Autenticación
- */user* - Usuarios
- */memory* - Juego (Memoria web)
 - */card* - Manejo de las cartas
 - */level* - Manejo de los niveles
 - */score* - Puntuaciones

Esta serie de rutas no representa una versión final de los módulos de ruta que contendrá el sistema, sino una representación de los posibles módulos que existirán dado los requerimientos actuales.

En Relación a la Base de Datos

Para el manejo de la base de datos, uno de los propósitos del sistema es que el almacenamiento de todos los datos estén en MySQL.

Sin embargo, como se mencionó anteriormente el actual sistema de Frontend utiliza PouchDB para el almacenamiento de los datos relacionados al juego. Esto conlleva a que han de realizarse cambios en la arquitectura del actual sistema de juego una vez completado el desarrollo del Backend.

A su vez, es de vital importancia para la sostenibilidad del proyecto el asegurar que exista correspondencia entre el actual esquema de datos que utiliza el Frontend y PouchDB y las tablas relacionales que contenga el Backend en



MySQL. Para esta correspondencia, se ha de utilizar las bases hechas por anteriores desarrolladores en temas de migración, para expandir e implementar de acuerdo al contexto actual del proyecto.

Proyecto

Roadmap

Para la implementación del proyecto, se tiene previsto determinar y representar los avances del sistema como “hitos” a cumplir en el desarrollo. A su vez, para medidas de identificación, estos hitos han de estar identificados con las respectivas versiones del sistema, siguiendo el esquema de versionamiento propuesto con anterioridad.

En base a esto, se propone de forma inicial, el siguiente ruta para el sistema:

1. **v0.0.0** - Creación del repositorio
2. **v0.1.0** - Inicialización de Node.js
3. **v0.2.0** - Implementación de la base de datos en MySQL
4. **v0.3.0** - Migración de modelos de PouchDB a MySQL
5. **v0.4.0** - Diseño del modulo de usuarios
6. **v0.5.0** - Módulo de autenticación y usuarios
7. **v0.6.0** - Módulo de Juegos

Cabe destacar que esta ruta no representa una versión final de los resultados, sino una aproximación a lo que podría resultar en base a los requerimientos y propuesta actual.

Recomendaciones

Una recomendación para el abordaje de este proyecto es la consideración de introducir la tecnología de Docker, para automatizar despliegues y pruebas en distintos equipos del sistema previamente descrito.

Docker es una herramienta de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.



El sistema puede verse ampliamente beneficiado del uso de Docker, dado que facilita los requerimientos para desplegar el proyecto o hacer pruebas de forma local en diferentes equipos ya que debido a su naturaleza de contenedor, asegura que el comportamiento e instalación sea consistente en todos los casos.