



Parte del Plan de Trabajo de Servicio Comunitario

Sistema ANAR

Manual de Desarrollo

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



Introducción

Propósito del Documento

Este documento servirá como documentación del proceso de desarrollo y especificaciones técnicas del Sistema ANAR y sus respectivas dependencias: módulo de autenticación, juego de Memoria Rupestre, y aplicaciones.

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, ahora que el proceso de desarrollo inicial ha concluido.

Objetivos del Producto Mínimo Viable (MVP)

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

- Tener un módulo de juego de memoria rupestre que sea capaz de suplir las necesidades y requerimientos de la respectiva aplicación.
- Contar con una implementación en base de datos MySQL que use de guía los modelos de entidad-relación creados para el módulo.
- El módulo debe ser independiente del resto de módulos del Sistema ANAR, tal que en caso de ser requerido pueda ser expandido en un futuro, y a su vez no tenga problema integrándose a cualquier aplicación que sea parte de ANAR.
- Para toda la información de un usuario relacionada al juego de memoria, debe de existir una conexión a nivel relacional con los modelos de usuario centrales del módulo de usuarios.
- Debe seguir el lineamiento de la arquitectura detallada en la propuesta principal.



Descripción

Descripción del Problema

Como fue descrito en el documento original de la propuesta Inicial de Servicio Comunitario, se está implementó un backend con API para centralizar todas las aplicaciones de ANAR tal que exista una especie de “centro de operaciones” donde la información de un usuario pueda servir para conectarte a los diferentes servicios que ofrece ANAR. A su vez, como primera aplicación desarrollada e integrada al Sistema consistirá en el Juego de Memoria Rupestre tanto en versión web como versión móvil.

De esa forma, para el manejo de este módulo ha de ser necesario que existan rutas y controladores capaces de administrar la información relacionada al juego de memoria y los elementos que lo componen.

Solución Implementada

La solución implementada consistió en un servicio de Backend en un entorno NodeJS, con los suficientes módulos iniciales como para mantener las funcionalidades (principalmente de juego y manejo de accesos) del juego “Memoria Rupestre” en su versión web, y futura actualización a versión móvil.

Para el manejo del juego de Memoria se tuvo bastante en cuenta todo el manejo de la información que compone las distintas partes del juego no puede ser cambiada por cualquier usuario y se introdujeron roles de usuarios a la aplicación a través del módulo de Autenticación con sus respectivas integraciones al módulos de aplicaciones (para manejar ciertos estados únicos de cada aplicación).

Sin embargo, lo más importante de este proyecto consistió 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.

Inconvenientes Presentados

- Como fue expuesto en la propuesta inicial y subsecuentes documentos, la cantidad de cambios necesarios para la obtener el MVP hizo que por fuera del alcance inicial la conexión/actualización en el Frontend de este backend.



Arquitectura

Descripción

Para la construcción de la aplicación, se realizó en distintas etapas de desarrollo siguiendo una pseudo-metodología ágil donde se definen una serie de tareas y responsabilidades dentro de un alcance específico, normalmente asociado con un módulo de funcionalidades en concreto.

Los módulos desarrollados fueron los siguientes:

- Autenticación
- Aplicaciones (Incluido conexión al módulo de autenticación con Permisos y Roles)
- Juego de Memoria Rupestre (Dificultad, Niveles, Puntuaciones y Trivia)

Todas las etapas de desarrollo fueron realizadas utilizando un entorno en NodeJS, base de datos relacional MySQL manejados mediante Sequelize en el entorno de desarrollo. Principalmente, para la construcción de controladores y endpoints siguiendo los principios de API REST se utiliza el framework Express.

Para más información sobre las tecnologías y metodologías utilizadas durante el desarrollo, se hace referencia al documento de Propuesta general. Ningún cambio mayor se realizó durante el proceso, únicamente se agregó la herramienta de Joi para la verificación de valores de entrada en los endpoints, de esta forma se puede tener un control detallado sobre los valores y tipos que pueden tener tanto en el cuerpo, cabecera o parámetro de la solicitud.

Todo el proyecto está almacenado utilizando el sistema de control de versiones Git, se posee un repositorio completo con todos los cambios realizados hasta la fecha almacenado usando el servicio de GitHub, este repositorio se describe en la siguiente sección.

Repositorio de Github

Descripción

El repositorio de GitHub asociado al proyecto pertenece a una organización denominada [SC-ANAR](#), que contiene todos los repositorios de otros proyectos relacionados a la organización.



El repositorio puede ser encontrado en el siguiente [link](#).

Las ramas del repositorio fueron manejadas usando la estrategia de *gitflow*. De esa forma, todos los cambios aprobados y principales se encuentran en la rama principal del repositorio: *main*.

El repositorio contiene un README que indica ciertas configuraciones de instalación y pruebas que también están especificados en este documento, pero de manera puntual. También se incluyen enlaces útiles y algunos ejemplos de configuración.

Instalación de Ambiente de Desarrollo

Requisitos de Uso

Requisitos de uso

Para poder ejecutar el proyecto, se debe de tener instalado en el equipo las siguientes dependencias de programas con sus respectivas versiones:

- node >= 16.0.0
- npm >= 7.14.0
- mysql == 8.0.31

Instalación de Dependencias

Una vez se cumplan los requisitos de instalación, se debe correr el siguiente comando para instalar todas las librerías que usa el programa para su ejecución:

```
npm install
# También es posible, si desea utilizar `yarn` como instalador
yarn install
```

Comandos Disponibles

El proyecto cuenta con una serie de comandos disponibles que el desarrollador puede utilizar. Todos los comandos deben llevar como prefijo la expresión:

```
npm run {{ COMANDO }}
```

Ahora, la lista de comandos disponibles incluye:



- `dev`: Es utilizado para crear un servidor de desarrollo que "escucha" los cambios en todos los archivos disponibles dentro del proyecto, para que cuando detecte un cambio automáticamente se reinicie sin necesidad de algún input extra.
- `build`: Crea una versión estática del proyecto en versión de producción. Esto implica que se hacen algunas optimizaciones que no están disponibles en el servidor de desarrolladores
- `start`: Comienza un servidor con la última versión del proyecto construida.

Si se va a desarrollar contenido nuevo para la aplicación, se recomienda encarecidamente utilizar: `npm run dev` como comando sobre el que trabajar.

Variables de Entorno

Para la utilización del Sistema ANAR, se hace uso de variables de entorno para las conexiones que dependan de información sensible como lo pueden ser credenciales de bases de datos, información de ambiente de desarrollo, clave de encriptación para autenticación, etc.

Al momento de instalar el proyecto, ninguna variable de entorno estará configurada. Sin embargo, dentro del proyecto se encuentra un archivo `.env.sample` donde se encuentra una **lista inicial de variables de entorno que necesita el proyecto para operar con normalidad.**

Archivos y Organización

Directorio

La estructura de directorios del proyecto se divide en dos carpetas principales:

- `config`

Dentro de este directorio únicamente se encuentra un archivo local donde se cargan todas las variables de entorno que se deben de cargar para satisfacer las necesidades del proyecto.

- `src`
 - `classes`

Se encuentran las declaraciones de interfaces, enumeradores y clases de objetos que se definieron fuera del alcance de los modelos de base de datos.



Estas clases o interfaces, pueden poseer sus métodos propios para manipulación de la información proveída.

- *controllers*

Los distintos archivos con la lógica de negocios para la implementación de los distintos módulos. Todo separado por módulos y sub-módulo en caso de ser necesario.

- *database*

Posee los archivos con las funcionalidades respecto a la configuración y utilización de todo lo relacionado a la base de datos. Igualmente, define las relaciones entre distintas tablas/modelos que puedan existir.

- *middleware*

Aquellas funciones de las que los controladores hacen uso para distintos objetivos comunes se pueden encontrar acá, como por ejemplo, la función de jerarquía mayor para la validación de data de entrada de los endpoints o la función que descifra los token JWT se pueden encontrar aca.

- *models*

Todas las tablas de base de datos se registran como clase/modelos dado el uso del ORM utilizado en el proyecto. La declaración e iniciación de esos modelos sucede en este directorio.

- *routes*

Posee todas las declaraciones de rutas y las llamadas a controladores de los endpoints. Todas están divididas por archivos que hacen referencia a los módulos o sub-módulos de los que se hacen llamados, por motivos de legibilidad.

- *scripts*

Actualmente posee únicamente un solo script, para hacer la primera inicialización de la base de datos cuando se instala por primera vez el proyecto.

- *utils*

Distintas funciones de utilidad que se utilizan a lo largo de diferentes módulos del proyecto.



- *validators*

Contiene todos los validadores que se utilizan en los endpoints para confirmar que todas las partes de la petición del API estén en orden.

Base de Datos

Definición

Para la creación de la versión final de la base de datos, se basó en el estudio y creación de varias iteraciones inicialmente de modelos Entidad-Relación (Extendidos), donde todas las partes interesadas pudieron refinar en base a los requerimientos los valores mínimos con el que una base de datos para todo el Sistema ANAR pudiera empezar a funcionar, teniendo en cuenta el alcance del proyecto general para toda la organización.

Toda la documentación respecto a cómo está estructurada y diseñada la base de datos puede encontrarse en los distintos documentos propuestas que fueron proporcionados con la entrega de la aplicación y desarrollado con esta igualmente.

Agregar un nuevo Módulo al Sistemas

Descripción

Se deben de seguir los siguientes pasos para expandir el sistema a nuevas aplicaciones:

1. Agregar la aplicación al enumerador de aplicaciones
2. Definir los respectivos roles y permisos correspondientes
 - a. Agregar luego su declaración al script de inicialización de la base de datos.
3. Crear los modelos de base de datos
 - a. Crear las relaciones entre modelos de base de datos
 - b. Agregar la inicialización al archivo dentro de ese mismo directorio
4. Crear los controladores con la respectiva lógica de utilización
 - a. Se recomienda que los controladores sigan como mínimo los estándares de CRUD (Create, Read, Update y Delete) para la utilización de datos.
 - b. Intentar mantener el código modularizado, para mejor escalabilidad en el futuro.



- c. Cualquier función de utilidad, se puede guardar en el directorio respectivo.
5. Crear los validadores de datos para la petición
6. Crear el archivo con la respectiva declaración de ruta
 - a. Agregar la ruta al módulo principal
7. ¡Listo! Disfrute de su nuevo endpoint
 - a. Como última recomendación, se sugiere actualizar la documentación con esta nueva información.