**PROYECTO DE PRÁCTICA: “Administrador de recibos”**

*miércoles 15 de mayo del 2024*

**Definición del proyecto**

**I.- Objetivo:**

Generar un proyecto de práctica en el cual se realiza una API y una página Web que se encarguen de administrar los recibos de un usuario, el usuario puede listar todos sus recibos, crear nuevos, editarlos y eliminarlos.

**II.- Requerimientos:**

1. Generar una Base de datos en la cual se almacenen los recibos generados por el usuario.
2. Generar un Proyecto API REST para la gestión de los servicios y su almacenamiento en base de datos.
3. Generar un Proyecto Web para mostrar la información al usuario y que pueda interactuar con la generación, edición y eliminación de los servicios; el proyecto web debe conectarse al proyecto API REST.

*Especificaciones Base de datos:*

* Deben generarse las tablas user, receipt, token.
* Todas las tablas deben tener los campos adicionales fecha creación y fecha actualización.
* En la tabla Usuario debe almacenarse la contraseña del usuario con formato hash.
* En la tabla user y receipt debe almacenarse la URL de la imagen cargada de la foto de perfil o del recibo almacenado.
* En la tabla Token deben almacenarse los tokens de una sesión activa.

*Especificaciones proyecto API REST:*

* Se debe generar un proyecto API REST que contenga servicios que realicen todas las funciones de un CRUD (Crear, Leer, Actualizar y Eliminar) con la información de los recibos, los recibos pueden o no tener una fotografía del recibo.
* Deben generarse los servicios para la creación de un usuario, su validación de inicio de sesión, el refrescamiento de token de sesión, cerrado de sesión y actualización de datos del usuario.
* Todos los servicios deben tener una respuesta estructurada con Código de respuesta, mensaje y los datos retornados.
* Todos los servicios deben estar protegidos con un JWT.
* Todos los servicios deben validar la información recibida por el usuario antes de interactuar con la base de datos.
* Deben generarse las pruebas unitarias y de integración de todos los servicios de la aplicación (Herramienta de pruebas Jest).
* Debe generarse la documentación de los servicios y los esquemas de consulta (Herramienta de desarrollo Swagger).
* Al crear o modificar datos de un usuario, la contraseña debe pasar por un proceso de hash para que esta se oculte en base de datos.
* Debe crearse un Bucket en AWS para el almacenamiento de imágenes de foto de perfil de usuario y foto de los recibos.

*Especificaciones proyecto Web:*

* Crear una pantalla para Inicio de sesión y Registro de nuevo usuario.
* Crear una pantalla dashboard en la cual se muestre una tabla que liste todos los recibos del usuario.
* En la tabla de la pantalla dashboard para cada registro debe mostrarse 3 botones en una columna: 1.- Ver información completa, 2.- Editar información y 3.- Eliminar información.
* Todas las pantallas deben contener un Header y un Footer que muestre la información del usuario, un botón de configuración y un botón se cerrar sesión.
* Crear una pantalla con un formulario en la cual se pueda crear un nuevo recibo. Esta pantalla debe accederse desde la pantalla dashboard, se debe validar la información antes de enviarla.
* Crear una pantalla con un formulario en la cual se pueda editar un recibo existente y debe de precargarse la información. Esta pantalla debe accederse desde la pantalla dashboard, se debe validar la información antes de enviarla.
* Debe generarse una pantalla configuración desde la cual se modifique la información del usuario, debe precargarse la información al ingresar a la pantalla, se debe validar la información antes de enviarla.
* En todas las pantallas debe haber un contador interno que revise el token de sesión del usuario, si está por expirar debe indicársele al usuario y refrescarse el token, y si ya expiró debe retornar al usuario a la pantalla Inicio de sesión.
* Todas las pantallas deben ser responsivas a tamaños grande (tamaño monitor pc), mediano (tamaño table) y pequeño (tamaño teléfono celular).

**III.- Tecnologías por utilizar:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Documentación** | **Base de datos** | **Almacenamiento** | **Backend** | **Frontned** |
| Ms Office Word | PostgreSQL | Bucket de AWS | Node.js | React.js |
|  |  | Git y GitHub | Express | React-router-dom |
|  |  | Render | Sequelize ORM | Axios |
|  |  | Vercel | JWT | React-hook-form |
|  |  |  | Bycrypt | SweetAlert |
|  |  |  | AWS S3 | FontAwesome |
|  |  |  | Jest | HTML, CSS y JavaScript |
|  |  |  | Swagger |  |
|  |  |  | Colors |  |
|  |  |  | CORS |  |
|  |  |  | Morgan |  |
|  |  |  | Validate |  |
|  |  |  | Supertest |  |

**IV.- Plan de actividades:**

DOCUMENTACIÓN INICIAL:

1.- Generar documento “Definición de proyectos”.

2.- Generar documento “Diagrama de base de datos”.

3.- Generar documento “Diseño de interfaces”.

BASE DE DATOS

4.- Crear una base de datos local ejecutando los querys correspondientes.

5.- Crear datos iniciales en base de datos local ejecutando los querys correspondientes.

DISEÑO DE PANTALLAS

6.- Diseñar pantalla Inicio de sesión y Registro nuevo usuario.

7.- Diseñar pantalla Dashboard.

8.- Diseñar pantalla Formulario Crear/Editar recibo.

9.- Diseñar pantalla configuración usuario.

GENERACIÓN DE BUCKET AWS

10.- Generación de Bucket de AWS para almacenamiento de foto de perfil de usuarios.

11.- Generación de Bucket de AWS para almacenamiento de recibos.

BACKEND

12.- Crear el proyecto “Server” con Node.js e instalar todas las dependencias de desarrollo.

13.- Generar archivos de configuración de conexión a base de datos, servidor y CORS.

14.- Generar archivos de modelos de tablas de base de datos.

15.- Generar archivos de middlewares para validación de tokens.

16.- Generar archivos de controlador para consulta a base de datos y retorno de información.

17.- Generar archivos de rutas para consulta de servicios (o endpoints).

18.- Generar archivos de utilerías para AWS S3, JWT y formateo de fechas.

19.- Generar archivos de pruebas con Jest.

20.- Generar el código para gestión de documentación con Swagger.

21.- Realizar pruebas unitarias y de integración con el código generado con Jest.

22.- Realizar pruebas de servicios utilizando la página del proyecto de Swagger.

23.- Generar documento “Realización de pruebas con Jest y Swagger” tomando screenshots de las pruebas con Jest y Swagger.

FRONTEND

24.- Crear el proyecto “Client” con React.js e instalar todas las dependencias de desarrollo.

25.- Crear pantalla Inicio de sesión/ Registro nuevo usuario y su funcionalidad.

26.- Crear pantalla Dashboard y su funcionalidad.

27.- Crear pantalla Formulario Crear/Editar recibo y su funcionalidad.

28.- Crear pantalla Configuración de usuario y su funcionalidad.

29.- Agregar funcionalidad para validación de token de sesión y refrescamiento de token.

30.- Realizar pruebas integrales de toda la aplicación para probar todas las pantallas y la conexión con los servicios de la API “Server”.

ALMACENAMIENTO DE PROYECTO

31.- Almacenar en GitHub el proyecto “Server”

32.- Almacenar en GitHub el proyecto “Client”

33.- Almacenar en GitHub toda la documentación generada.

DESPLIEGUE DE PROYECTO

34.- Crear en Render la Base de datos de PostgreSQL de recibos para el proyecto.

35.- Almacenar y publicar en Render el proyecto API REST “Server” y configurar las variables de entorno para la conexión.

36.- Almacenar y publicar en Vercel el proyecto Web “Client” y configurar las variables de entorno para la conexión.

37.- Agregar en el GitHub de los proyectos un archivo README.md en el cual se coloquen las URLs de los proyectos publicados, las instrucciones de descarga de los proyectos y base de datos, configuración de los proyectos, y ejecución de proyectos en entorno local.

**V.- Tiempo estimado de desarrollo:**

Tiempo: 75 hrs

Involucrado: Desarrollador Jr. - Alan Eduardo Aguilar Guerrero

“El proyecto estará depositado en el GitHub con su respectiva documentación”.