Definición:

Para el desarrollo backend del proyecto tiendas de barrio virtuales se decidió utilizar Ruby on Rails (RoR) el cual es un framework para el desarrollo de aplicaciones web de código abierto escrito en el leguaje de programación Ruby, este se destaca por estar enfocado en la productividad, la simplicidad y la promoción de mejores prácticas de desarrollo, está diseñado para facilitar la programación de aplicaciones, ya que permite escribir menos código a comparación de otros frameworks y promueve que el enfoque este dado a la funcionalidad de los sistemas.

Ventajas:

* Don´t repeat yourself : DRY es una primicia que parte de afirmar que cada conocimiento debe tener una representación única, al promover no escribir una y otra vez el mismo código hace que la escalabilidad, el mantenimiento y la productividad al momento de desarrollar sean mayores con respecto a otros frameworks.
* Convention Over Configuration: Rails utiliza una serie de convenciones de forma predeterminada que evita que se especifiquen o se tome mucho tiempo en codificar cosas que en la práctica se vuelven irrelevantes como lo son los interminables archivos de configuración que la mayoría de frameworks hay que realizar uno por uno.
* Arquitectura MVC: Rails sigue el patrón de modelo-vista-controlador, lo cual facilita la organización y el mantenimiento del código gracias a la estructuración.
* Generadores de código: Rails proporciona generadores automáticos que permiten crear rápidamente partes comunes de la aplicación con una sola línea de comando lo cual acelera el proceso de desarrollo.
* Gemas: Rails utiliza gemas las cuales son librerías o extensiones que facilitan incorporar funcionalidades a la aplicación.
* Enfoque RESTful: Rails promueve la creación de aplicaciones web siguiendo los principios REST, lo cual facilita la construcción y utilización de APIs.
* Seguridad: Rails por defecto cuenta con protección de falsificación de solicitudes entre sitios e inyecciones de SQL.
* Escalabilidad y productividad: Rails es adecuado para grandes y pequeñas aplicaciones ya que tiene una gran capacidad de agregar funcionalidades y servicios según sean necesarios, del mismo modo permite la alta productividad al enfocarse en la creación de código de valor para las aplicaciones.

Instalación:

Para utilizar RoR se debe instalar en el equipo ruby y Sqlite como base, primero se procede a instalar de la pagina <https://rubyinstaller.org/> Ruby+Devkit 3.2.2-1 (x64)

Texto, Sitio web

Descripción generada automáticamente

Posteriormente se ejecuta el archivo y se aceptan los terminos de licencia se agrega ruby al path para poder utilizarlo desde la consola de windows

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

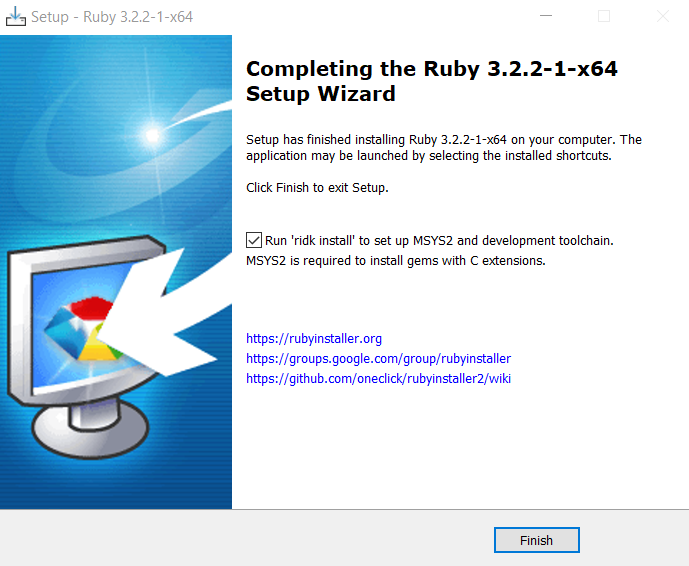
Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

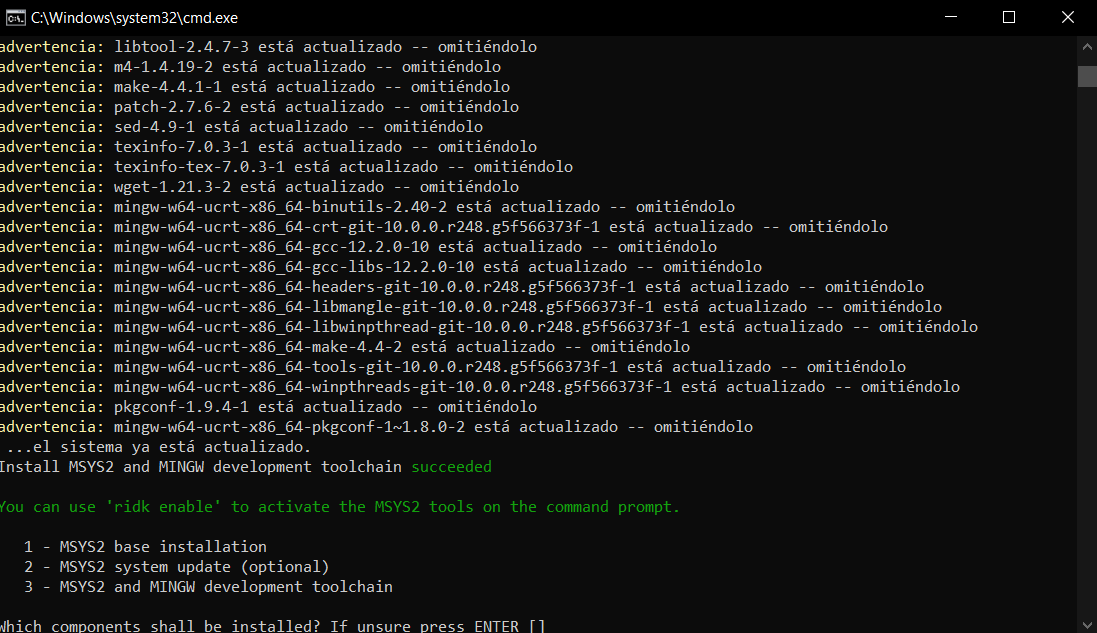
Descripción generada automáticamente Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Una vez finalizada la instalación se ejecuta el el ridk install y se instala MSYS2, ya instalado en un terminal se digita un comando para comprobar la versión de ruby

 Texto

Descripción generada automáticamente

 Texto

Descripción generada automáticamente

Posteriormente se debe instalar el framework con el siguiente comando

Texto

Descripción generada automáticamente

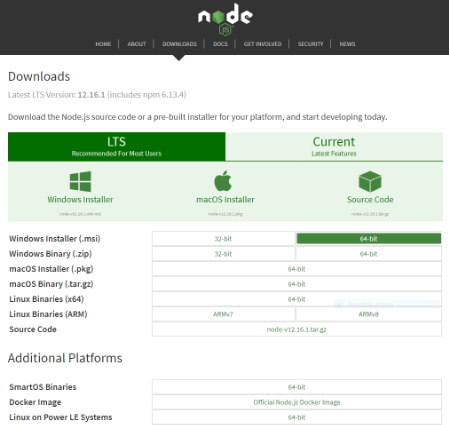
Se verifica la version de rails y se procede a instalar la gena Bundler la cual maneja las dependencias de Ruby

Imagen que contiene Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

De la misma manera deben instalar Node.js https://nodejs.org/en/download y Yarn https://classic.yarnpkg.com/en/docs/install/#windows-stable

 Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

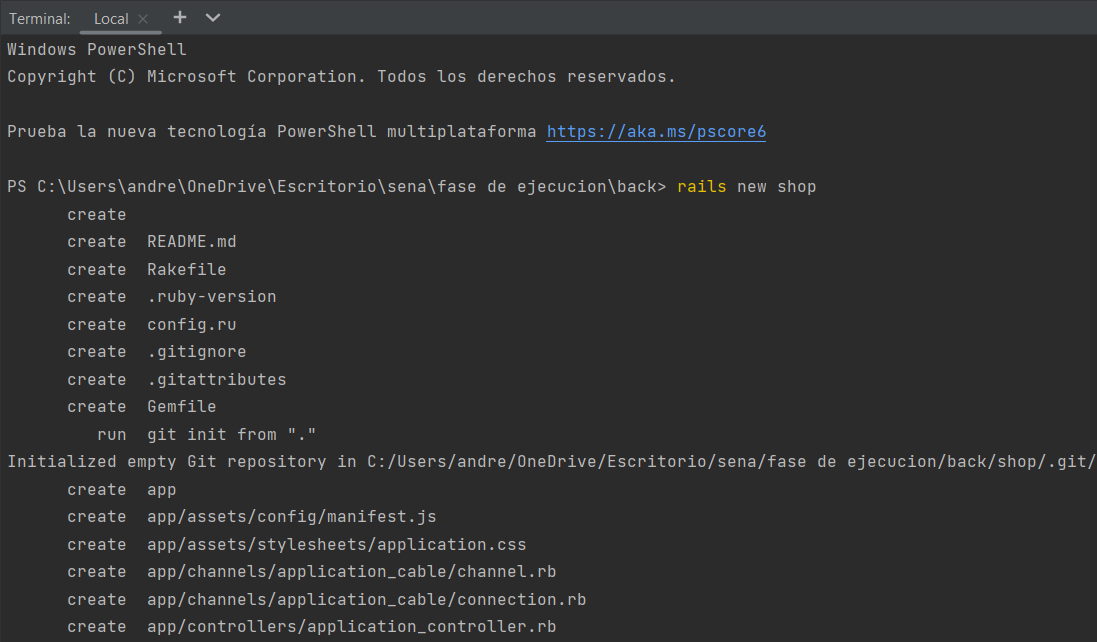
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Sitio web

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Creación de aplicación:

Rails tiene una serie de scripts generadores, que están diseñados para facilitar el desarrollo y poder comenzar a trabajar en tareas particulares, uno de ellos es el comando rails new, el cual crea una base para la nueva aplicación con los archivos necesarios en el directorio actual del terminal.



Al terminar la ejecución se genera una carpeta con toda la base del proyecto

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Tabla de archivos principales y su función:

|  |  |
| --- | --- |
| Archivo/Carpeta | Objetivo |
| app/ | Contiene los controladores, modelos, vistas, ayudas, anuncios, canales, trabajos y activos de la aplicación. |
| bin/ | Contiene diferentes scripts de Rails que inician la aplicación, del mismo modo otros scripts para configurar, actualizar, implementar o ejecutar la aplicación |
| config/ | Contiene la configuración para las rutas, la base de datos, entre otros archivos importantes de configuración. |
| db/ | Contiene el esquema de la base de datos y sus migraciones |
| Gemfile/ | Permite especificar las dependencias de gemas que son necesarias para la aplicación de Rails. |
| lib/ | Contiene los módulos extendidos de la aplicación. |
| log/ | Contiene los archivos de registro de aplicaciones. |
| public/ | Contiene los archivos estáticos y compilados. |
| Rakefile | Localiza y carga las tareas que se pueden ejecutar desde el cmd. |
| README.md | Es el manual de instrucciones de la aplicación. |
| storage/ | Archivos de almacenamiento activo para Disk Service. |
| test/ | Contiene las pruebas unitarias, accesorios y aparatos de pruebas en general |
| tmp/ | Archivos temporales como el cache. |
| vendor/ | Se almacena el código de terceros. |
| .gitattributes | Define metadatos para rutas especificas en el repositorio git |
| .gitignore | Le dice a git que archivos debe ignorar. |
| .ruby-version | Contiene la versión predeterminada de Ruby. |

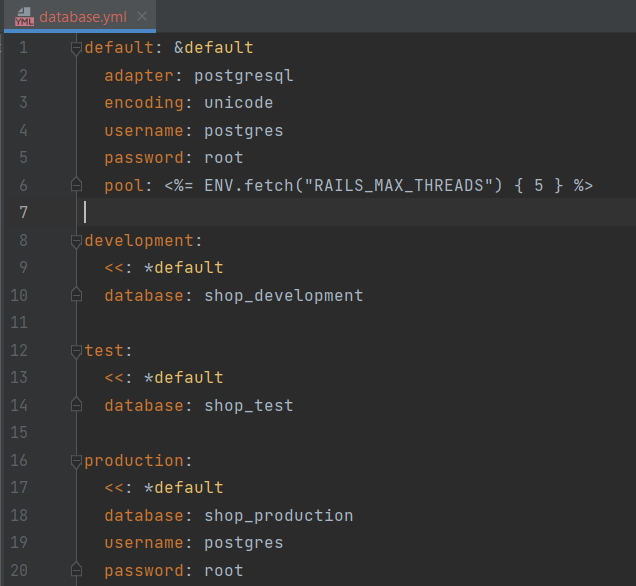
Configuración de la base de datos:

Se decide trabajar con PostgreSQL ya que tiene mejor compatibilidad con Rails, primero se crean diferentes bases de datos de la tienda para producción, desarrollo y pruebas en la herramienta pdAdmin 4

Interfaz de usuario gráfica

Descripción generada automáticamente

Estas bases de datos se deben configurar en el archivo config/database.yml del proyecto con sus respectivas credenciales



Prueba de servidor local:

Una vez configurada la base de datos con el comando rails s, se levanta el servidor local el cual apunta a <http://127.0.0.1:3000/> en el cual se harán pruebas a medida que se vaya desarrollando la aplicación, así se comprueba que la instalación y la creación base de la aplicación fue correcta.

Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Creacion de tabla de Productos:

Se genera el modelo para la tabla de productos definiendo los parametros y los tipos de datos que tendra con el comando rails generate model Product name:string description:text stock:integer price:decimal rank:integer idCategory:integer idProduct:integer

Posteriormente se realiza una migración para la creacion de esta tabla con el comando rails db:migrate

Texto

Descripción generada automáticamente

Esto crea el modelo en la carpeta models/product.rb

Interfaz de usuario gráfica, Texto, Aplicación, Teams

Descripción generada automáticamente

Se genera el controlador con el comando rails generate controller Products y se agrega la ruta en el archivo config/routes resources :products , se utiliza resaurces para generar rutas RESTful estandar

Texto

Descripción generada automáticamente

Obteniendo el archivo app/controllers/products\_controller.rbInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Se procede a implementar las acciones esandar para operaciones CRUD en el controlador

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Explicación de código:

class ProductsController < ApplicationController

Esto define la clase ProductsController, que es una subclase de ApplicationController. Los controladores en Rails suelen heredar de ApplicationController para heredar funcionalidades comunes.

before\_action :set\_product, only: [:show, :edit, :update, :destroy]

Esta línea utiliza el filtro before\_action para ejecutar el método set\_product antes de ciertas acciones. En este caso, se aplica antes de las acciones show, edit, update, y destroy. El propósito de este filtro es encontrar el producto correspondiente basado en el parámetro :id antes de ejecutar estas acciones.

Index (Listar Productos)

Propósito: La acción index se utiliza para mostrar una lista de todos los productos disponibles.

Lo que hace: En esta acción, generalmente recuperas todos los registros de productos desde la base de datos (por ejemplo, @products = Product.all) y los pasas a la vista para que se muestren.

Show (Mostrar Detalles de Producto)

Propósito: La acción show se utiliza para mostrar los detalles de un producto específico.

Lo que hace: Recupera un producto en función de su identificador único (generalmente el ID) y lo pasa a la vista correspondiente para mostrar sus detalles.

New (Crear Nuevo Producto)

Propósito: La acción new muestra un formulario para crear un nuevo producto.

Lo que hace: En esta acción, generalmente creas una nueva instancia del modelo de producto (@product = Product.new) que se utilizará para el formulario en la vista. Esto permite al usuario ingresar detalles del nuevo producto.

Create (Guardar Nuevo Producto)

Propósito: La acción create se utiliza para procesar y guardar un nuevo producto en la base de datos.

Lo que hace: Recibe los datos enviados desde el formulario en la vista (generalmente a través de params) y crea un nuevo registro de producto en la base de datos. Luego, redirige al usuario a la página de detalles del producto o a la lista de productos.

Edit (Editar Producto)

Propósito: La acción edit muestra un formulario para editar un producto existente.

Lo que hace: Recupera un producto en función de su identificador único (generalmente el ID) y lo pasa a la vista del formulario de edición. Esto permite al usuario modificar los detalles del producto.

Update (Actualizar Producto)

Propósito: La acción update se utiliza para procesar y guardar las modificaciones realizadas en un producto.

Lo que hace: Recibe los datos enviados desde el formulario de edición y actualiza el registro de producto correspondiente en la base de datos con los nuevos valores. Luego, redirige al usuario a la página de detalles del producto actualizado.

Destroy (Eliminar Producto)

Propósito: La acción destroy se utiliza para eliminar un producto de la base de datos.

Lo que hace: Recupera un producto en función de su identificador único (generalmente el ID) y lo elimina de la base de datos. Luego, redirige al usuario a la lista de productos actualizada.

def set\_product

Este es un método privado que encuentra el producto correspondiente en la base de datos utilizando el :id pasado como parámetro en la URL. Por ejemplo, si la URL es /products/1, este método buscará el producto con el ID 1 en la base de datos y lo asignará a la variable de instancia @product. Esto permite reutilizar el código para encontrar productos en múltiples acciones.

def product\_params

Este es otro método privado que se utiliza para definir los parámetros permitidos que pueden enviarse desde un formulario. En Rails, es una práctica común proteger tu aplicación de la asignación masiva de atributos (mass-assignment) permitiendo solo parámetros específicos. En este caso, product\_params permite los atributos :name, :description, y :price para el modelo Product. Esto es importante para la seguridad y previene ataques de asignación masiva no autorizada.

Generación de la vista de productos:

Con el comando rails generate erb:scaffold Product se genera los archivos de cada una de las vistas .html.erb en el directorio app/views/products.

Texto

Descripción generada automáticamente

Ahora se configura la vista inicial para poder ver los productos listados en el index, se procede a realizar una tabla que sea representativa por el momento para posteriormente agregar todos los modelos realizados en el front, se realiza un bucle para llamar a todos los productos en la base de datos, se poscisionan en la tabla y organizan las diferentes acciones de la siguiente manera:

Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Del mismo modo se realiza la vista para agregar un nuevo producto con sus respectivos campos de formulario:

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

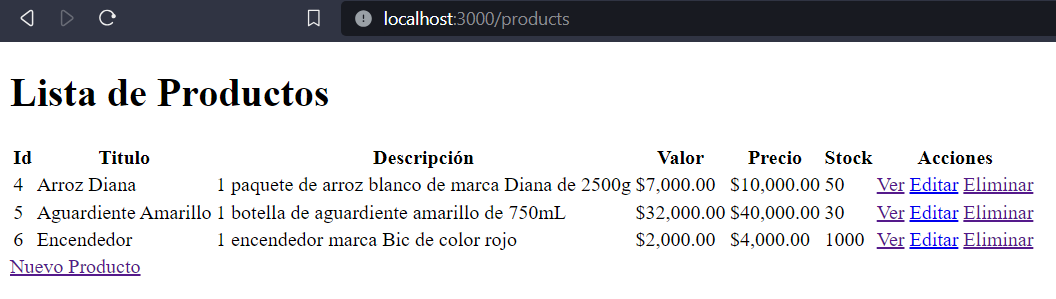
Se procede a realizar pruebas e ingresar productos por medio de la interfaz ya que se tiene configurado lo necesario

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Se realiza a inserción de varios productos:



Se configura la vista para editar productos.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

Se realizan pruebas de edición:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Pruebas de eliminación del producto con id 6:

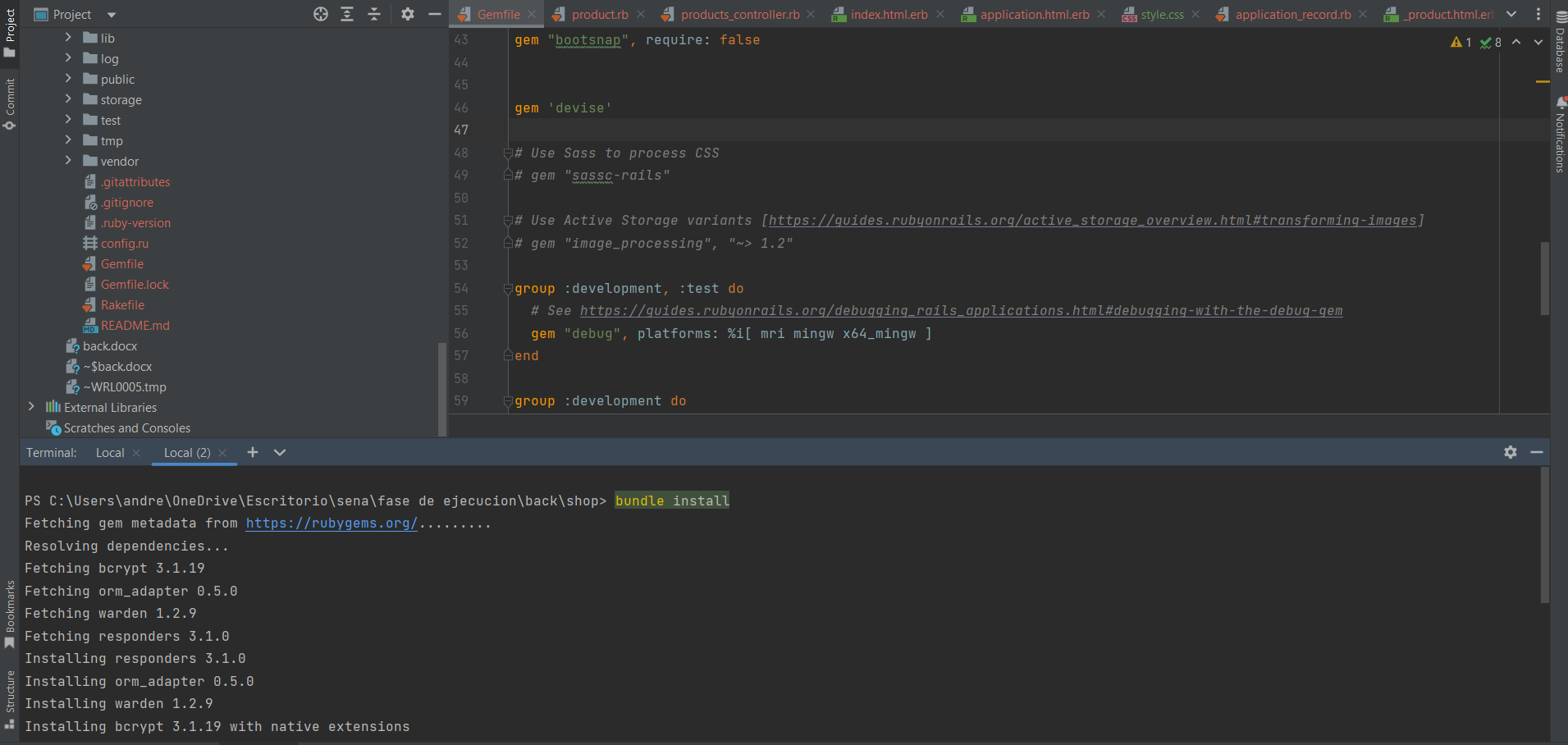
Texto

Descripción generada automáticamente con confianza media

Creacion de usuarios con autenticación

Para crear un modelo de usuario con autenticación en una aplicación Ruby on Rails, es común utilizar una gema llamada "Devise". Devise es una gema muy popular que proporciona una solución completa de autenticación de usuarios, incluyendo registro, inicio de sesión, recuperación de contraseña, entre otros.

Se agrega la gena devise y se realiza un bundle install para instalar las dependencias:



Se procede a generar el modelo de Usuario y a realizar la migración para aplicar los cambios a la base de datos

Texto

Descripción generada automáticamente

Se gereran las vistas con el comando rails generate devise:views

Texto

Descripción generada automáticamente

Se crean una serie de vistas configurables para distintas acciones de autenticación Texto

Descripción generada automáticamente

En el archivo de turas config/routes se agregan configuraciones de la navegación

Texto

Descripción generada automáticamente

Configuración de atributos en el registro:

Con el comando rails generate devise:controllers users -c=registrations, se genera un controlador de registro para configurar los datos que se solicitaran al registro en la aplicación

Texto

Descripción generada automáticamente

En el archivo creado registrations\_controller se configuran los campos que se desean permitir en el registro de usuario definiendo el metrodo sign\_up\_params

Texto

Descripción generada automáticamente

y se deben agregar al modelo estos nuevos atributos con el comando rails generate migration AddAttributesToUsers name:text phone:integer address:string fiscal\_number:integer y luego se ejecuta una migración

Texto

Descripción generada automáticamente

Ahora se procede a generar esos campos en la vista de registro

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Tabla

Descripción generada automáticamente con confianza baja

Configuración de sesiones

Se genera el controlador para configurar los inicios de sesión con el comando rails generate devise:controllers users/sessions el cual crea una serie de archivos configurables para las sesiones

Texto

Descripción generada automáticamente con confianza media

Se configura el sessions\_controller.rb para que vaya a la pagina de inicio de forma predeterminada cuando exista un registro

Texto

Descripción generada automáticamente

Se configura la vista de login:

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Se modifica el archivo de rutas con las respectivas acciones personalizadas de registros, sesiones y la pagina de inicio de la aplicacion

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Se configura la pagina de inicio