Relatorio - RideFast API

Informacoes do Projeto

Disciplina: Programacao Funcional
Professor: Bruno Lopes
Equipe: Fernando e Alberto

Video de Demonstracao

https://youtu.be/Epeqm--uFnk

Tecnologias Utilizadas

- **Framework:** Phoenix (Elixir)
- **Banco de Dados:** MySQL
- **Autenticacao:** JWT (Guardian)
- **Hash de Senhas:** BCrypt

Endpoints Implementados (35 total)

Autenticacao (2)

Metodo	Rota	Descricao
POST	/api/v1/auth/register	Registrar usuario ou motorista
POST	/api/v1/auth/login	Login

Usuarios (5)

Metodo	Rota	Descricao
GET	/api/v1/users	Listar usuarios
GET	/api/v1/users/:id	Buscar usuario
POST	/api/v1/users	Criar usuario
PUT	/api/v1/users/:id	Atualizar usuario
DELETE	/api/v1/users/:id	Deletar usuario

Motoristas (5)

Metodo	Rota	Descricao
GET	/api/v1/drivers	Listar motoristas
GET	/api/v1/drivers/:id	Buscar motorista
POST	/api/v1/drivers	Criar motorista
PUT	/api/v1/drivers/:id	Atualizar motorista
DELETE	/api/v1/drivers/:id	Deletar motorista

Perfil do Motorista (3)

Metodo	Rota	Descricao
GET	/api/v1/drivers/:id/profile	Buscar perfil
POST	/api/v1/drivers/:id/profile	Criar perfil
PUT	/api/v1/drivers/:id/profile	Atualizar perfil

Veiculos (4)

Metodo	Rota	Descricao
GET	/api/v1/drivers/:id/vehicles	Listar veiculos do motorista
POST	/api/v1/drivers/:id/vehicles	Adicionar veiculo
PUT	/api/v1/vehicles/:id	Atualizar veiculo
DELETE	/api/v1/vehicles/:id	Deletar veiculo

Idiomas (5)

Metodo	Rota	Descricao
GET	/api/v1/languages	Listar idiomas
POST	/api/v1/languages	Criar idioma
GET	/api/v1/drivers/:id/languages	Listar idiomas do motorista
POST	/api/v1/drivers/:id/languages	Adicionar idioma ao motorista
DELETE	/api/v1/drivers/:id/languages/:lang_id	Remover idioma do motorista

Corridas (10)

Metodo	Rota	Descricao
GET	/api/v1/rides	Listar corridas
POST	/api/v1/rides	Criar corrida
GET	/api/v1/rides/:id	Buscar corrida
DELETE	/api/v1/rides/:id	Deletar corrida
POST	/api/v1/rides/:id/accept	Aceitar corrida
POST	/api/v1/rides/:id/start	Iniciar corrida
POST	/api/v1/rides/:id/complete	Finalizar corrida
POST	/api/v1/rides/:id/cancel	Cancelar corrida
GET	/api/v1/rides/:id/history	Historico da corrida
POST	/api/v1/rides/:id/ratings	Avaliar corrida

Avaliacoes (1)

Metodo	Rota	Descricao
GET	/api/v1/drivers/:id/ratings	Listar avaliacoes do motorista

Implementacao JWT (Guardian)

Arquivo: `lib/ride_fast_api/auth/guardian.ex`

```elixir
defmodule RideFastApi.Auth.Guardian do
  use Guardian, otp_app: :ride_fast_api

  alias RideFastApi.Users
  alias RideFastApi.Drivers

  def subject_for_token({:user, user}, _claims) do
    {:ok, "user:#{user.id}"}
  end

  def subject_for_token({:driver, driver}, _claims) do
    {:ok, "driver:#{driver.id}"}
  end

  def subject_for_token(_, _) do
```

```elixir
    {:error, :invalid_role}
  end

  def resource_from_claims(claims) do
    case claims["sub"] do
      "user:" <> id ->
        case Users.get_user(id) do
          nil -> {:error, :user_not_found}
          user -> {:ok, {:user, user}}
        end

      "driver:" <> id ->
        case Drivers.get_driver(id) do
          nil -> {:error, :driver_not_found}
          driver -> {:ok, {:driver, driver}}
        end

      _ ->
        {:error, :invalid_subject}
    end
  end
end
```

Implementacao BCrypt

Arquivo: `lib/ride_fast_api/users/user.ex`

```elixir
defmodule RideFastApi.Users.User do
  use Ecto.Schema
  import Ecto.Changeset

  @derive {Jason.Encoder, only: [:id, :name, :email, :phone]}
  schema "users" do
    field :name, :string
    field :email, :string
    field :phone, :string
    field :password_hash, :string
    field :password, :string, virtual: true

    has_many :rides, RideFastApi.Rides.Ride

    timestamps(type: :utc_datetime)
  end

  def changeset(user, attrs) do
```

```
    user
    |> cast(attrs, [:name, :email, :phone, :password])
    |> validate_required([:name, :email, :phone, :password])
    |> validate_format(:email, ~r/^[^\s]+@[^\s]+$/, message: "deve ser um
email valido")
    |> unsafe_validate_unique([:email], RideFastApi.Repo, message: "ja esta em
uso")
    |> put_password_hash()
  end

  defp put_password_hash(%Ecto.Changeset{valid?: true, changes: %{password:
password}} = changeset) do
    put_change(changeset, :password_hash, Bcrypt.hash_pwd_salt(password))
  end

  defp put_password_hash(changeset), do: changeset
end
```

Maquina de Estados das Corridas

Arquivo: `lib/ride_fast_api/rides.ex`

```
defmodule RideFastApi.Rides do
  import Ecto.Query, warn: false
  alias RideFastApi.Repo
  alias RideFastApi.Rides.Ride

  def accept_ride(%Ride{} = ride, driver_id, vehicle_id) do
    if ride.status == "SOLICITADA" do
      ride
      |> Ride.changeset(%{driver_id: driver_id, vehicle_id: vehicle_id,
status: "ACEITA"})
      |> Repo.update()
    else
      {:error, :invalid_status}
    end
  end

  def start_ride(%Ride{} = ride) do
    if ride.status == "ACEITA" do
      ride
      |> Ride.changeset(%{status: "EM_ANDAMENTO", started_at:
NaiveDateTime.utc_now()})
      |> Repo.update()
    else
```

```
      {:error, :invalid_status}
    end
  end

  def complete_ride(%Ride{} = ride, final_price) do
    if ride.status == "EM_ANDAMENTO" do
      ride
      |> Ride.changeset(%{status: "FINALIZADA", final_price: final_price,
endend_at: NaiveDateTime.utc_now()})
      |> Repo.update()
    else
      {:error, :invalid_status}
    end
  end

  def cancel_ride(%Ride{} = ride) do
    if ride.status in ["SOLICITADA", "ACEITA", "EM_ANDAMENTO"] do
      ride
      |> Ride.changeset(%{status: "CANCELADA"})
      |> Repo.update()
    else
      {:error, :invalid_status}
    end
  end
end
```

```
SOLICITADA -> ACEITA -> EM_ANDAMENTO -> FINALIZADA
     |          |           |
     v          v           v
  CANCELADA  CANCELADA    CANCELADA
```

Como Executar

1. Instalar dependencias:

```
mix deps.get
```

2. Criar e migrar banco:

```
mix ecto.create
mix ecto.migrate
```

3. Iniciar servidor:

```
mix phx.server
```

A API estara disponivel em `http://localhost:4001`