

uso_da_ia

leia todo o projeto, e monte um arquivo md com toda a sintaxe necessária para fazer ele, sem usar exemplos reais, use exemplos semelhantes da sintaxe para que em duvida eu olhe

- Vou analisar todo o projeto e criar um guia de sintaxe completo.
 - Read(/media/fernando-bezerra/Files1/CODES/Elix/AV3/RideFast-API/[AV3] - Rest API com Phoenix.pdf)
 - └ Read PDF (448.1KB)
 - Read(/media/fernando-bezerra/Files1/CODES/Elix/AV3/RideFast-API/ridefast-rest-api.yaml)
 - └ Read 1067 lines
 - Agora vou criar o guia de sintaxe completo:
 - Write(guia_sintaxe_phoenix.md)
 - └ Wrote 994 lines to guia_sintaxe_phoenix.md
- ```
Guia de Sintaxe - Phoenix API REST
Este guia contem toda a sintaxe necessaria para criar uma API REST com
Phoenix, Ecto, Guardian (JWT) e BCrypt.

1. ESTRUTURA DO PROJETO
...
meu_projeto/
```

... +984 lines (ctrl+o to expand)

- Arquivo criado: Alberto/guia\_sintaxe\_phoenix.md

O guia contém toda a sintaxe com exemplos genéricos para:

1. Estrutura do projeto
2. mix.exs - Dependências (Phoenix, Ecto, Guardian, BCrypt)
3. config.exs - Configuração do Guardian e banco
4. Migrations - Criar tabelas, índices, foreign keys, tabelas N:N
5. Schemas - Básico, com password/BCrypt, tabela N:N
6. Contexts - CRUD completo, máquina de estados

7. Guardian - JWT (subject\_for\_token, resource\_from\_claims)
8. Auth Pipeline e Error Handler
9. Router - Pipelines, rotas públicas/protegidas, rotas aninhadas
10. Controllers - CRUD, autenticação, máquina de estados, rotas aninhadas
11. Códigos HTTP - :ok, :created, :not\_found, etc.
12. Validações - validate\_required, validate\_format, unique\_constraint
13. Queries Ecto - where, join, preload
14. Comandos mix - create, migrate, phx.server
15. Pattern matching e pipe operator
16. Resumo de arquivos necessários

me dê um regex que aceita um ou mais caracteres não-espço, seguido de @, seguido de um ou mais caracteres não-espço

● <sup>[1]</sup>+@\s+\$

Em Elixir:

~r/<sup>[2]</sup>+@\s+\$/

me dê um diagrama ASCII de máquina de estados: SOLICITADA vai para ACEITA, que vai para EM\_ANDAMENTO, que vai para FINALIZADA. Todos os três primeiros estados podem ir para CANCELADA

● SOLICITADA → ACEITA → EM\_ANDAMENTO → FINALIZADA

|||

v v v

CANCELADA CANCELADA CANCELADA

Analise o código e crie um txt com os payloads de request do projeto, para que eu possa usar no postman

● Read(lib/ride\_fast\_api\_web/controllers/auth\_controller.ex)

└ Read 171 lines

● Read(lib/ride\_fast\_api\_web/controllers/ride\_controller.ex)

└ Read 176 lines

● Read(lib/ride\_fast\_api\_web/controllers/rating\_controller.ex)

└ Read 51 lines

● Write(postman\_payloads.txt)

└ Wrote 478 lines to postman\_payloads.txt

=====

=====  
=====

## RideFast API - Payloads para Postman

Base URL: http://localhost:4001/api/v1

=====

====

=====

IMPORTANTE: Para endpoints autenticados, adicionar header:

Authorization: Bearer {TOKEN}

=====

====

=====

## AUTENTICACAO (2)

... +468 lines (ctrl+o to expand)

● Arquivo criado: Alberto/postman\_payloads.txt

Contém todos os 35 endpoints com:

- URL completa
- Método HTTP
- Headers necessários
- Body JSON de exemplo
- Indicação [AUTH] para endpoints que precisam de token
- Fluxo completo de teste sugerido
- Diagrama da máquina de estados

---

1. ^\s ↩

2. ^\s ↩