**Elixir – Paradigma funcional**

Cadmiel Matioli Donato

RESUMO

A partir do estudo realizado no elixir a ideia principal será conhecer a linguagem, o que se pode fazer com ela, sua história e sua estrutura, assim agregando conhecimento a partir de pesquisas e realizando a implementação dela na pratica, entendi que é possivel encontrar modelos de linguagens que trabalham de uma forma completamente diferente do convencional que vemos logo quando começamos a estudar programação. Com tudo que vemos durante o tempo de estudo da para entender que é possivel desenvolver aplicações escalaveis a partir de uma linguagem funcional tornando uma aplicação de alta qualidade.

**Palavras-chave:** Elixir, Programação, Funcional, MVC

**Elixir - Functional paradigm**

ABSTRACT

From the study carried out in the elixir, the main idea will be to know the language, what can be done with it, its history and its structure, thus adding knowledge from research and implementing it in practice, I understood that it is possible to find models of languages ​​that work in a completely different way than the conventional ones we see right when we start to study programming. With everything that we see during the time of study, to understand that it is possible to develop scalable applications from a functional language making it a high quality application.

**Key-words:** Elixir, Programming, Functional, MVC

1. **INTRODUÇÃO**

A programação funcional no elixir não é um código matemático absoluto com apenas funções recursivas ou cheio de caracteres desconhecidos e aglomerados. O principal objetivo da programação funcional é compor funções reduzidas e imutáveis e funções que mantêm seu estado e não se alteram. Um modelo de programação no elixir é utilizar o pipe operador, que ajuda muito na hora de construir uma função com imutabilidade. Comparando com a orientação a objetos vemos uma diferença principal com relação a programação funcional a qual na orientação a objetos temos na chamada um objeto e respectivamente seu método, já na programação funcional temos um modulo e respectivamente suas funções.

1. **SOBRE O ELIXIR**

Uma linguagem de programação Brasileira criada por José Valim, ela utilizar o paradigma funcional que por sua vez é executado em uma máquina virtual conhecida como Erlang, o elixir é utilizado na web, sistemas embarcados, ingestão de dados e processamento de multimidia, algumas aplicações que são conhecidas e utilizam a linguagem é o Whatsapp, Discord, adobe entre outros, uma de suas vantagens é a escalabilidade, como todas as linguagens, o elixir também tem seus pontos negativos como o processamento computacional pesado ela não tem um suporte muito bom para isso como python por exemplo, o mesmo ocorre com deep learning e no desenvolvimento de apps.

1. **IMUTABILIDADE**

Ao passar um valor para uma variável é necessário instanciar novamente pois os valores são armazenados em posições de memória diferentes, fazendo com que não seja armazenado o estado da função. Isso serve para que quando for realizar procedimentos assíncronos não seja pesado ou dê algum problema pois geralmente nesses casos não há garantia da integridade do valor salvo em memória.

1. **ONDE MAIS PODEMOS ENCONTRAR FUNÇÕES DE ALTA ORDEM**

Programação funcional aparece em outras linguagens de uma forma diferente, mas existe, como no caso de um map, filter ou reduce em JavaScript, onde se passa uma função dentro de outra função, recebendo o nome de função de alta ordem.

1. **SISTEMA DE ARQUIVOS**

No elixir utilizando o Phoenix vi que tem uma estrutura de arquivo que lembra um pouco o laravel + django para quem conhece esses frameworks vai se sentir familiarizado.

Pois vemos que existe a pasta assets e static onde comporta-se os arquivos de manipulação visual e importações para o projeto funcionar corretamente estilizado, também vemos a pasta config onde dá para se conectar as aplicações externas como banco de dados, as portas de conexão e por sua vez os arquivos que são gerados e carregados quando se utiliza o npm, deps e \_build, são diretórios onde vamos encontrar as dependências e bibliotecas utilizadas na aplicação que são geradas, não elas não sobem para o git assim como a node\_modules pois são arquivos que são utilizados localmente, conhecidos como arquivos de build do projeto.

A pasta lib pode ser um tanto quanto confusa, pois é lá que está o projeto e não as bibliotecas, lá vamos encontrar a controller, schemas, templates, views, todos esses arquivos vão compor desde a geração do banco de dados até a interface visual, somando tudo isso podemos chamar de um sistema MVC.

Dentro da pasta priv temos as migrations que foram geradas automaticamente dentro dela também tem um item que pode ser muito útil ao decorrer do projeto para facilitar nos testes ou na criação de usuários administradores que é o seeder, esse item gera valores no banco de dados, ele pode inserir tantos dados aleatórios quanto dados estáticos como no caso de um sistema que se inicia com apenas um usuário default.

Test é uma pasta onde vamos encontrar o que compõe um sistema de teste, nele é possível criar o que é chamado de ambiente de teste, que com isso podemos criar cenários perfeitos ou não para uma determinada ação da aplicação que resultara em uma reação, isso quer dizer que por exemplo, ao clicar em um botão ele deve abrir uma determinada pagina, caso abra e execute tudo corretamente é definido o que se espera, o esperado é que a pagina abra na rota correta, caso não ocorra essa situação que é esperada pelo teste o teste falhara.

Em seguida temos outros dois arquivos que vão compor o projeto sendo o formater que formatará o código e o mix que realizara toda a manipulação da aplicação, ou seja um gerenciador de pacotes como npm, manage.py ou php artisan.

1. **INTERAÇÃO COM O USUÁRIO**

Ao utilizar o gerador do Phoenix ele criou um CRUD que já vinha composto de um dashboard, com isso vemos alguns dados da máquina como processos e versões que estão sendo executadas naquele momento, em relação ao CRUD, começando pela criação dos dados na tabela, vemos que tem uma tela de criar e inserir os campos de forma bem básica, nada além de um formulário com alguns estilos, em seguida temos a tela de detalhes, editar e deletar, a tela principal para manipular esses itens é a listagem o que é chamado de index.

1. **RESULTADOS ADQUIRIDOS**

No elixir a linguagem se comporta de maneira diferente das outras linguagens como vimos acima e na prática, mas ainda assim conseguimos manter um conceito MVC por exemplo, no qual utilizei o modelo no vídeo, dessa forma é possível realizar chamadas de módulos e suas funções de maneira semelhante as outras linguagens “mais convencionais”, ou seja, é mais tranquilo de se aprender seu comportamento, ao construir o mini CRUD com dados de um sistema bancário utilizando o Phoenix entendi como conceitualizar a ideia, pois ela tem em suas bibliotecas frameworks para web, IOT entre outros, e esses frameworks alguns básicos já vem instalado o que por sua vez facilita na sua utilização dependendo do projeto em que está sendo trabalhado. Entendi como passar parâmetros sendo eles até mesmo funções dentro de seus modelos, e realizar a geração de valores para banco, testes, geração do próprio CRUD, iniciar a aplicação e manipulação dos estilos e interface gráfica(front-end).

1. METODOLOGIA

Utilizei dos meios de sites como blogs, Youtube e artigos para ter como embasamento a criação do projeto e o entendimento dele, a partir do estudo desse material consegui acumular conhecimento suficiente para poder criar toda a estrutura e corrigir os erros ao longo da aplicação

REFERÊNCIAS

RocketSeat. Elixir - O que é programação funcional? - Code/drops #73. Youtube, 31 mai. 2021. Disponível em: <https://youtu.be/53Lv3efp7Rk>. Acesso em: 05 set. 2021

RocketSeat. Elixir - Quais as diferenças entre Elixir e Ruby? | Como aprender Elixir do jeito certo? - #PR 46. Youtube, 28 jul. 2021. Disponível em: <https://youtu.be/53Lv3efp7Rk>. Acesso em: 05 set. 2021

RocketSeat. Phoenix - Crie um CRUD em minutos com Phoenix Generators - Code/drops #78. Youtube, 14 jun. 2021. Disponível em: <https://youtu.be/JwOE5gLlwZc>. Acesso em: 05 set. 2021

Elixir, c2021. Página inicial. Disponível em: <https://elixir-lang.org/>. Acesso em: 05 de nov. de 2021.

Phoenix, c2020. Página inicial. Disponível em: <https://hexdocs.pm/phoenix/installation.html>. Acesso em: 05 de nov. de 2021.

Santos, Peterson. FAQ Elixir: Tudo o que você precisa saber para fazer a escolha certa. 02 jun. 2020. Disponível em: <https://ateliware.com/blog/tudo-que-voce-precisa-para-escolher-elixir>. Acesso em: 06 de nov. de 2021.

PetSI. ELIXIR: uma linguagem de programação brasileira em sistemas distribuídos do mundo. 14 set. de 2018. Disponível em: <http://www.each.usp.br/petsi/jornal/?p=2459>. Acesso em: 06 de nov. de 2021.