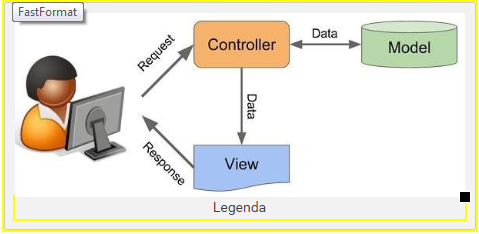
1. **Introdução**

O Trabalho em Questão é referente a um sistema de Apostas desenvolvido a nível de Desktop. O Sistema sempre será utilizado apenas por um usuário, e será sempre o mesmo. A Necessidade se deu visto a necessidade de Organização do processo, e de automatização do mesmo. O Sistema não se mostrou ser essencial, mas definitivamente encessário visto a grande quantidade de pessoas a participar dos bolões da Empresa.

1. **Camadas de Aplicação**

Será dividido em 3 camadas, model, view e controller(MVC):



1. **Linguagem de Programação**

Devido à escrita derivada do C e por ser transpilado em javascript, foi escolhido a linguagem typescript.

O TypeScript possibilita que você escreva código JavaScript na forma que foi acostumado quando aprendeu Orientação a Objetos. Você lembra dessas aulas, em criar métodos que retornassem um valor com tipo definido, em criar classes e mais classes para o seu programa, em criar interfaces para desacoplar tudo que quisesse, entre diversas outras técnicas. Com TypeScript tudo isso é possível, porque no final ele pega o seu lindo código cheio de classes e transforma em JavaScript puro, no qual o browser vai compreender. No próprio site é definido que o TypeScript compila para JavaScript, o que é um termo tecnicamente errado – mas cada vez mais aceito, dado que aquele código “malucão” em JavaScript pode ser encarado como código de máquina.

1. **Ambiente.**

Devido à familiaridade da equipe e aos requisitos do sistema, foi escolhido o prórprio Desktop, visto que não há necessidade de integração Web

1. **Framework**

Electron.js, para gerar o aplicativo desktop(Windows/Linux) com code base em JavaScript

Electron (anteriormente conhecido como Atom Shell[2]) é um framework de código aberto criado por Cheng Zhao, e agora desenvolvido pelo GitHub[3]. Ele permite desenvolver aplicações para desktop GUI usando componentes front end e back end originalmente criados para aplicações web: Node.js para o back end e Chromium para o front end. Electron é o principal framework por trás de vários projetos notáveis de código aberto, incluindo Atom[4] da GitHub e os editores de código-fonte: Visual Studio Code[5] da Microsoft e o Light Table IDE, além da aplicação desktop do serviço de bate-papo Discord[6].

1. **Conexão**

Docker:

O Docker possibilita o empacotamento de uma aplicação ou ambiente inteiro dentro de um container, e a partir desse momento o ambiente inteiro torna-se portável para qualquer outro Host que contenha o Docker instalado.

Isso reduz drasticamente o tempo de deploy de alguma infraestrutura ou até mesmo aplicação, pois não há necessidade de ajustes de ambiente para o correto funcionamento do serviço, o ambiente é sempre o mesmo, configure-o uma vez e replique-o quantas vezes quiser.

Outra facilidade do Docker é poder criar suas imagens (containers prontos para deploy) a partir de arquivos de definição chamados Dockerfiles (veremos isso melhor em posts futuros).

Não podemos nos esquecer também de que o Docker utiliza como backend default o LXC, com isso é possível definir limitações de recursos por container (memória, cpu, I/O, etc.)

Para ficar um pouco mais claro, podemos ver na imagem abaixo a diferença em se utilizar o LXC puro e o Docker.

1. **Banco de Dados**

MySQL, pois é bastante conhecido pela equipe e possui geração de script baseado na modelagem gráfica do banco

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface. É atualmente um dos sistemas de gerenciamento de bancos de dados mais populares[1] da Oracle Corporation, com mais de 10 milhões de instalações pelo mundo.[2]

Entre os usuários do banco de dados MySQL estão: NASA, Friendster, Banco Bradesco, Dataprev, HP, Nokia, Sony, Lufthansa, U.S. Army, U.S. Federal Reserve Bank, Associated Press, Alcatel, Slashdot, Cisco Systems, Google, entre outros.

1. **Testes Unitários/Performance**

Karma, é possível fazer testes unitários e de Performance, rodando eles por essa biblioteca javascript.

Karma é um test runner feito para o AngularJs. O principal objetivo do Karma é automatizar os testes em diversos navegadores web com um único comando. Mesmo ele tendo sido criado para o AngularJs, atualmente ele é usado em outros frameworks JavaScript.

O Karma suporta diversos tipos de testes, como: unitários, integração, E2E.

1. **Testes Funcionais**

Spectron.js, possibilita testar a aplicação javascript pelos requisitos funcionais.