

**CURSO DE CIÊNCIA DA COMPUTAÇÃO**

**LINGUAGENS DE PROGRAMAÇÃO PARA WEB**

**PROF. MS. ANDERSON BRILHADOR**

**21 OUT. 2018**

**GABRIEL LENIN SILVA LIMA**

**RAFAEL BONIOLO**

**WILLIAN DE OLIVEIRA**

**SILVA**

**Trabalho:** Projeto 2 entrega 1

**Capitulo 1**

## **DOCUMENTAÇÃO DE REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS**

**Requisitos funcionais:**

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF001** | O sistema deve permitir ao usuário realizar depósitos e saques. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF002** | O sistema deve permitir opções de prazo mínimos para saque. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF003** | Caso o usuário saque antes do prazo cobrar uma taxa administrativa de 0,5%. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF004** | O sistema deve permitir gerar relatórios de rendimentos para que o usuário (investidor) acompanhe os seus ganhos. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF005** | O sistema deve permitir que os usuários (gestor de investimento) faça investimentos com o dinheiro depositado pelos investidores. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF006** | O sistema deve permitir que o usuário (gestor de investimentos) tenha acesso ao dashboard de ações. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF007** | O sistema deve permitir que o usuário (gestor de investimentos) acompanhe a progressão de ganhos da sua carteira de investimentos. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF008** | O sistema deve permitir que o usuário (administrador) realize o cadastro de novos usuários (administrador, gestor e investidor). | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF009** | O sistema deve permitir ao administrador definir um valor limite para compra de investimentos (gestor), sendo sempre menor que o valor de dinheiro em caixa e definir a meta de rendimento mensal para cada gestor. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF010** | O sistema deve permitir que o usuário (administrador) acompanhe a progressão de ganhos da empresa (fluxo de caixa), por gestor e por investidor. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF011** | O sistema deve conter vários filtros para visualização dos dados como por período, por investimentos e etc... | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF012** | O sistema deve ter uma porcentagem máxima para o fundo de investimento, e o restante deve ser destinado ao fundo de saques. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RF013** | Se não houver nenhum cadastro de usuários (investidor, gestor ou administrador) no banco de dados o sistema deve permitir a criação de um usuário administrador. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

**Requisitos não funcionais:**

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RNF001** | Botões de compra e venda de fácil acesso. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RNF002** | O sistema deve ter disponibilidade em 99,9% do tempo. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RNF003** | O sistema deve ter desenvolvimento em linguagem de programa web PHP 7. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RNF004** | O sistema deve rodar em um hardware com no mínimo: processador i3, RAM de 4GB e memória de 500GB. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RNF005** | Os relatórios não podem exceder o prazo de 5 segundos para serem gerados. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RNF006** | Deve ser possível a atualização do sistema sem precisar pará-lo. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Prioridade** |
| **RNF007** | O sistema deve possuir um layout amigável e responsivo. | **Importante** |
| **Versão** | **Fonte** |
| **1** | Anderson Brilhador |

**Capitulo 2**

**ARQUITETURAS E TECNOLOGIAS QUE SERÃO UTILIZADAS**

Como arquitetura geral do software será utilizada a arquitetura cliente/servidor. A tecnologia cliente/servidor é uma arquitetura na qual o processamento da informação é dividido em módulos ou processos distintos. Um processo é responsável pela manutenção da informação (servidores) e outros responsáveis pela obtenção dos dados (os clientes). Os processos cliente enviam pedidos para o processo servidor, e este por sua vez processa e envia os resultados dos pedidos. É no servidor que normalmente ficam os sistemas mais pesados da rede, tais como o banco de dados. As máquinas clientes são menos poderosas, pois não rodam aplicativos que requerem tantos recursos das máquinas. Optou-se por usar cliente-servidor para dinamizar a aplicação, e deixando fácil a inclusão de uma versão mobile, desktop, ou até outra tecnologia web.

O cliente é a parte que interage com o usuário, possui a interface que o usuário utiliza para requisitar as tarefas ao servidor, sendo chamado de front-end da aplicação. Os processos clientes gerenciam as atividades dos usuários e realizam as validações dos dados informados por estes. É o processo que responde a uma mensagem solicitando a realização de alguma tarefa por parte do cliente. O processo servidor é chamado de back-end. O processo servidor pode oferecer serviços a muitos clientes, realiza pesquisas, filtragens e atualizações em bancos de dados. Esta arquitetura juntamente com o modelo MVC foram escolhidos, pois proporcionam uma maior flexibilidade do sistema, ou seja, torna o código mais dinâmico.

Em paralelo ao cliente/servidor optou-se por utilizar o padrão MVC, devido á arquitetura que possui, o que possibilita a divisão do projeto em camadas muito bem definidas. Desta forma será possível encapsular o model e o controller dentro do servidor, e a view por sua vez no cliente, para realizar a gestão do front-end.

A utilização do padrão MVC traz como benefício isolar as regras de negócios da lógica de apresentação, a interface com o usuário. Isto possibilita a existência de várias interfaces com o usuário que podem ser modificadas sem que haja a necessidade da alteração das regras de negócios, proporcionando assim muito mais flexibilidade e oportunidades de reuso das classes.

Uma das características de um padrão de projeto é poder aplicá-lo em sistemas distintos. O padrão MVC pode ser utilizado em vários tipos de projetos como, por exemplo, desktop, web e mobile.

Reflection (em português: reflexão) é o processo em que um programa pode observar e modificar sua própria estrutura, no PHP este recurso foi adicionado após a reformulação da orientação a objeto na linguagem, versão 5, possibilitando realizar engenharia reversa em classes, interfaces, funções e extensões, além disso também podemos resgatar comentários/documentação de funções, classes e métodos. A capacidade de executar/invocar métodos dinamicamente utilizando a reflexão é justamente a principal causa pela decisão de usar este processo. Desta forma é possível desenvolver um software de maneira dinâmica.

O Angular é um framework JavaScript que simplifica não apenas a construção da interface de usuário, mas também o desenvolvimento de aplicações client-side diferenciadas, sejam elas para a web, mobile ou desktop. O Angular é Plataforma de aplicações web de código-fonte aberto e front-end, será utilizado em conjunto com PrimeNG para a apresentação de gráficos e estatísticas na tela.

O PrimeNG possui cerca de 80 componentes de código aberto e gratuito com o uso sob a licença MIT. Foi desenvolvido pela PrimeTek Informatics e todos seus widgets estão hospedados no GitHub e a grande maioria são nativo. O PrimeNG é uma coleção de componentes para o Angular, todas as suas ferramentas são de código aberto e livre uso, será utilizado para melhor apresentação de conteúdos na tela.

O ECMAScript (ES) é a especificação da linguagem de script que o JavaScript implementa, ou seja, é a descrição formal e estruturada de uma linguagem de script, sendo padronizada pela Ecma International – associação criada em 1961 dedicada à padronização de sistemas de informação e comunicação – na especificação ECMA-262. No dia 17 de junho de 2015, foi definida a sexta edição da especificação, a ES6 (também chamada de ECMAScript 2015). Diferentemente das edições anteriores, o ES6 trouxe a maior mudança para a linguagem JavaScript desde a sua criação, há 20 anos. O principal objetivo da nova versão especificação foi tornar a linguagem mais flexível, enxuta e fácil de se aprender e trabalhar, tornando-a mais próxima a outras linguagens orientadas a objeto, como Java e Python.

A comunicação do cliente com o servidor é feita através do JSON (JavaScript Object Notation). Basicamente o JSON é um formato leve de troca de informações/dados entre sistemas, ou seja, é uma forma de serializar o objeto para a comunicação.

Exemplo:

Figura 1: Estruturação do arquivo no formato JSON.

Toda a estrutura do front-end será desenvolvida com base nessas quatro tecnologias: Angular, PrimeNG, ECMAScript e Bootstrap, todos serão utilizadas para facilitação no desenvolvimento da interface com o usuário, toda essa estrutura irá ser desenvolvida na parte cliente da arquitetura cliente/servidor. Serão utilizados estes 3 frameworks (Angular, PrimeNG e Bootstrap) pois fornecerão uma maior facilidade na criação de um design padrão e responsivo. Ou seja, através de suas documentações é possível reutilizar e agilizar todo o processo de desenvolvimento da interface do sistema.

Na figura 2 é possível entender basicamente como será a estrutura do sistema, apresentando a aplicação e organização de cada padrão e tecnologia dentro do software.

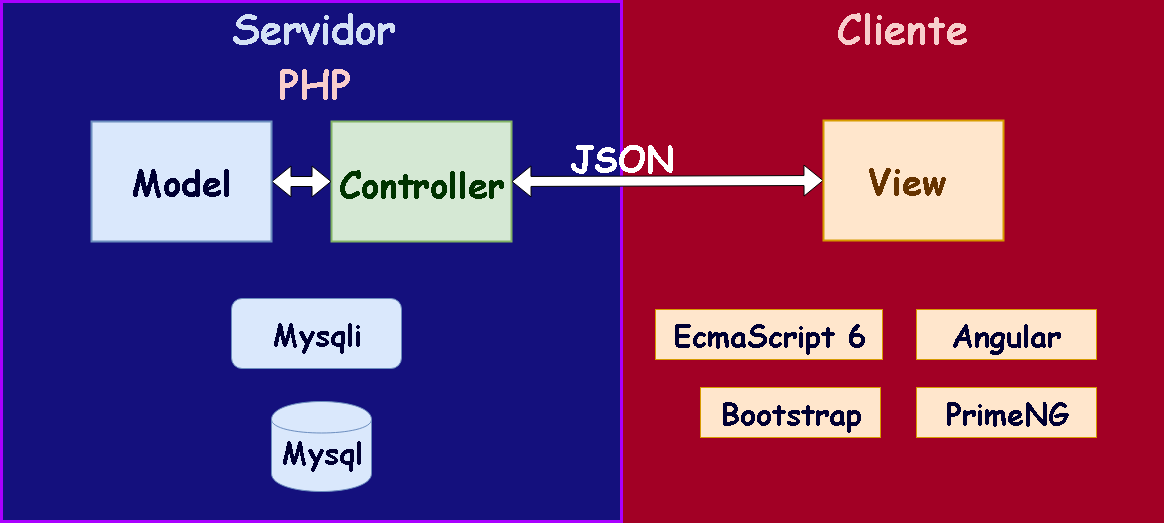


Figura 2: Visão macro da estrutura geral do sistema.

O MySQL Workbench é uma ferramenta de design de banco de dados visual que integra desenvolvimento, administração, design de banco de dados, criação e manutenção de SQL em um único ambiente de desenvolvimento integrado para o sistema de banco de dados MySQL. O MySql Workbanch será utilizado para a modelagem do banco de dados. O Git é um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de software, mas pode ser usado para registrar o histórico de edições de qualquer tipo de arquivo. Será utilizado para controle de versionamento do projeto, pois facilita o compartilhamento de um arquivo, auxiliando no controle de versão e segurança do arquivo.

GitHub é uma plataforma de hospedagem de código-fonte com controle de versão usando o Git. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo. Será utilizado para gerenciar os códigos do projeto, que pode ser encontrado no repositório.

O Waffle, se que conecta com o github, será utilizado para gerenciar o projeto, sua função é dividir as atividades para cada membro do grupo e acompanhar o desenvolvimento da atividade, e assim saber se a atividade está em progresso, em teste ou saber se a tarefa está terminada.

O PHP é uma linguagem de script usada para o desenvolvimento de aplicações web. O PHP 7 é uma versão superior do PHP 5, que traz melhorias como aumento de desempenho, tratamento de erros fatais, novos operadores, classes anônimas, suporte a Unicode, retiradas de funções entre outros.

O Bootstrap é um framework web com código-fonte aberto para desenvolvimento de componentes de interface e front-end para sites e aplicações web usando HTML, CSS e JavaScript. Utilizaremos o Bootstrap para desenvolver uma interface padrão e expansiva, para facilitar a construção da parte gráfica do sistema.

O MySQLi é uma maneira de acessar um servidor de banco de dados MYSQL. O MySQLi oferece uma API processual, que facilita a compreensão dos novos usuários. Optamos por utilizar o MySQLi pois facilita a conexão com o banco de dados, e facilita a criação de instruções das operações de consultas no Banco de Dados.

Relatórios de atividades:

|  |  |  |
| --- | --- | --- |
| **Alunos** | **Atividade** | **Importância** |
| Rafael | Criação da classe de conexão com banco de dados | Alta |
| Gabriel, Willian, Rafael | Adição de requisitos na lista de requisitos | Alta |
| Rafael | Atualização de requisitos não funcionais para funcionais | Alta |
| Gabriel, Willian | Adição de requisitos não funcionais adicionais | Alta |
| Gabriel | Formalização do documento de requisitos | Média |
| Willian | Revisão de requisitos | Alta |

Tabela 1: Relatório de atividade de entrega 1.

|  |  |  |
| --- | --- | --- |
| **Alunos** | **Atividade** | **Importância** |
| Rafael | Modelagem dos metodos de CRUD da conexão | Alta |
| Rafael | Modo de conexão dinâmica via reflexão | Alta |
| Gabriel | Leitura de configuração dinâmica do banco de dados | Alta |
| Rafael | Seleção da arquitetura, tecnologias e padrões | Alta |
| Gabriel, Willian | Pesquisa sobre a arquitetura | Alta |
| Gabriel, Willian | Pesquisa sobre as tecnologias e padrões | Alta |
| Gabriel, Willian, Rafael | Justificativa do uso da arquitetura, tecnologias e padrões | Alta |
| Willian | Formalização do documento de entrega | Média |

Tabela 2: Relatório de atividade de entrega 2.

Referencias:

GRUPO 4. **Como funciona a arquitetura cliente servidor**. Disponível em: <https://arqserv.wordpress.com/2012/03/17/como-funciona-a-arquitetura-cliente-servidor/>. Acesso em: out. 2018.

FILETO, R. **Sistemas-cliente servidor**. Disponível em: < http://www.inf.ufsc.br/~r.fileto/Disciplinas/BD-Avancado/Aulas/03-ClienteServidor.pdf> Acesso em: out. 2018.

HIGOR. **Introdução ao padrão MVC**. Disponível em: <https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308> Acesso em: out. 2018.

DEVMEDIA. **MongoDB, Express, Angulaar e Node.js**.Disponível em: <https://www.devmedia.com.br/guia/mongodb-express-angular-e-node-js/34007> Acesso em: out. 2018.

MILISE, A. **Então o que é ES6**. Disponível em: <https://desenvolvedor.expert/o-que-eh-es6-66c8d7631a0b> Acesso em: out. 2018.

LIMA, M. **O guia do ES6: tudo o que você precisa saber** <https://medium.com/@matheusml/o-guia-do-es6-tudo-que-voc%C3%AA-precisa-saber-8c287876325f> Acesso em: out. 2018.

PINHO, D. **O ECMAScript 6 e o futuro do javascript**. Disponível em: <https://imasters.com.br/front-end/o-ecmascript-6-e-o-futuro-do-javascript> Acesso em: out. 2018.

WIKIPEDIA. **MySQL Workbench**. Disponível em: <https://en.wikipedia.org/wiki/MySQL\_Workbench> Acesso em: out. 2018.

WIKIPEDIA. **GIT**. Disponível em: <https://pt.wikipedia.org/wiki/Git> Acesso em: out. 2018.

WIKIPEDIA. **GITHUB**. Disponível em: <https://pt.wikipedia.org/wiki/GitHub> Acesso em: out. 2018.

PLANROCKR. **Gerenciando projetos com Github Issue e Waffle.io**. Disponível em: <https://blog.planrockr.com/gerenciando-projetos-com-github-issues-e-waffle-io-8ad111213540> Acesso em: out. 2018.

WIKIPEDIA. **Bootstrap (framework front-end)**. Disponível em: <https://pt.wikipedia.org/wiki/Bootstrap\_(framework\_front-end)> Acesso em: out. 2018.

KEREZI, G. **Conheça as 7 maiores mudanças do PHP**. Disponível em: <http://blog.geekhunter.com.br/conheca-as-7-maiores-mudancas-do-php-7/> Acesso em: out. 2018.

AHMAD, H. **Diferenças – MySQLi versus PDO versus MySQL**. Disponível em: **<**https://imasters.com.br/back-end/diferencas-mysqli-versus-pdo-versus-mysql-benchmark-para-comparacao-de-desempenho-seguranca-e-conversor-que-funciona-em-2016-e-2017> Acesso em: out. 2018.

CROCKFORD. **Introducing JSON**. Disponível em: <https://www.json.org> Acesso em: out. 2018.

PRIMENG. **Why PrimeNG?**. Disponível em: < https://www.primefaces.org/primeng/#/> Acesso em: out. 2018.

ANGULAR. **One framework. Mobile & desktop**. Disponível em: < https://angular.io/> Acesso em: out. 2018.