INVEXT

DOCUMENTO DE ARQUITETURA DE SOFTWARE

1. INTRODUÇÃO

- - - - X

1. Finalidade

Este documento oferece uma visão geral da arquitetura abrangente do sistema, usando diversas visões arquitetônicas para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões ao sistema significativo que foram tomadas em relação ao projeto sobre a carteira digital.

2. ESCOPO

Esse documento serve de guia para outros objetivos de construção software em questão, a partir de um projeto desenvolvido envolvendo outros softwares e plataformas possíveis, a documentos possíveis da construção do projeto, onde é possível ter uma visão de cada tema.

3. CONFIGURAÇÕES, ACRÔNIMOS E ABREVIAÇÕES

a. API: É um acrônimo para Application Programming Interface(Interface de programação de aplicações). A API é um conjunto de definições e protocolos usados no desenvolvimento e na integração de um software, permitindo a interação com outros produtos sem a necessidade de interação com outro software.
b. UML: É um acrônimo para Unified Modeling Language(linguagem de modelagem unificada). O UML é uma linguagem utilizada para visualizar, especificar, construir e documentar a arquitetura completa de um software, fornecendo informações detalhadas para o desenvolvedor implementar o software.

4. REFERÊNCIAS BIOGRÁFICAS

Arquitetura de Software. Disponível em: http://repositorio.aee.edu.br/bitstream/aee/1106/3/TCC2_2018_2_GabrielLeiteDias_MatheusLimadeAlbuquerque_Apêndice2.pdf . Acesso em: 30 de abril de 2021.

O que é UML e Diagramas de Caso de Uso: Introdução Prática à UML. Disponível: https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408 . Acesso em: 03 de maio de 2021.

ReactJS Disponível em: https://pt-br.reactjs.org/ Acesso em: 03 de maio de 2021.

Node.JS Disponível em: https://nodejs.org/en/about/ Acesso em: 03 de maio de 2021.

5. VISÃO GERAL

O documento de arquitetura está organizado em informações da seguinte maneira:

i. Introdução.
ii. Escopo.
iii. Metas e Restrições de Arquitetura.
iv. Visão dos Casos de Uso.
v. Visão Lógica.
vi. Visão de Processos.
vii. Visão de Implantação.
viii. Visão de implementação.
ix. Tamanho e Desempenho.
x. Qualidade.

6. REPRESENTAÇÃO ARQUITETURAL

a. A Parte Dianteira:
    i. React: É um conjunto de bibliotecas de código aberto seguro para a criação de UIs interativas de forma mais fácil. Toda lógica é escrita em JavaScript da Repassagem de dados ao longo da passagem de dados. Essa tecnologia foi escolhida por quantidade de conteúdo disponível na internet, para o treinamento e aprendizado da equipe, outro ponto positivo é o fórum de dúvidas acerca do assunto estar sempre a disposição.
b. Parte Interna:
    i. Node.Js: O Node.Js é uma tecnologia usada para executar código em JavaScript fora do navegador. Com ele podemos construir aplicações web em geral, desde websites até APIs e microsserviços. Isso é possível graças a união do ambiente de execução do JavaScript pelo próprio motor de interpretação Node.Js e execução do JavaScript presente no Google Chrome chamado V8.

7. METAS E RESTRIÇÕES DA ARQUITETURA

a. Restrições:
    1. Software deve ser desenvolvido nas tecnologias;
    2. O ambiente de desenvolvimento do software deve funcionar tanto em windows, linux e MacOS;
    3. Para usar o Software é necessário Internet;
    4. O escopo final deve ser concluído até o final da disciplina;
b. Metas:

1. **Usabilidade** - O Software deve possuir alta aprendizagem e performance para atender os requisitos elicitados pelo grupo;
2. **Manutenção** - O código e as documentações realizadas devem estar num nível de qualidade, seguindo os padrões do projeto e bibliografia, onde sua manutenção seja fácil de ser realizada.

8. VISÃO DE CASOS DE USO

Para representar os Casos de Uso do sistema especificado, foi criado um diagrama de casos de uso que exibe os pontos principais do sistema.

9. VISÃO LÓGICA

A visão lógica descreve como o sistema é compatível, em termos de unidade e implementação. Mostra como é uma organização conceitual do sistema em termos de camadas, pacotes, classes e interfaces. O relacionamento entre os elementos mostra como dependências e interface, os relacionamentos parte assim por diante.

a. Diagrama de Aulas
   O diagrama representa como as classes serão programadas, os principais objetos ou realmente como diagrama completo pode ser encontrado na parte de Diagrama de Classes na wiki do projeto.
b. Diagrama de Pacotes
   O diagrama de pacotes é um diagrama estático que possibilita uma organização mais adequada ao sistema que representa uma versão de pacotes. O diagrama completo pode ser encontrado na parte de Diagrama de Pacotes da wiki do projeto.

c. Diagrama de Comunicação

O diagrama de comunicação mostra partes de comunicação entre objetos e/ou (representadas pela lifelines usando mensagens sequenciadas um arranjo) de forma livre. O diagrama completo pode ser encontrado na parte de Diagrama de Comunicação na wiki do projeto.

10. VISÃO DE PROCESSOS

a. Visão geral:

De tempo-de-execução como o sistema de execução de tempo-de-execução é construído na forma de um conjunto de tempo-de-execução que tem como modelo de comportamento de execução. Uma estrutura de tempo-de-execução normalmente tem semelhança com uma estrutura de código. Consistência de redes de comunicação rápida de objetos de comunicação.

b. Diagrama de Sequência:

O diagrama de sequência é uma das soluções fornecidas pela UML, que descrevem quimicamente o ciclo de vida do sistema em desenvolvimento. Descrição detalhada. O foco principal deste diagrama é descrito como troca entre os componentes do sistema e módulos que existem de uma maneira determinada e mensagens entre si. Os ciclos de vida podem ser aulas, atores ou mesmo abstrações que ocorrem entre aulas.

11. VISÃO DE IMPLANTAÇÃO

Descreve como a aplicação é disponibilizada para uso, seja em um ambiente de desenvolvimento, para testes ou em produção.

12. VISÃO DE IMPLEMENTAÇÃO

a. Visão Geral:

   Descreve como os defensores do desenvolvimento estão organizados no sistema de arquivos. Os elementos são arquivos e pastas(Quaisquer itens de configuração). Isto inclui as propriedades de desenvolvimento e os riscos de implantação.

b. Diagramas de Aulas:

   É uma representação da estrutura e relações das classes que servem de modelo para os objetos. Consiste em um conjunto de objetos com as mesmas características. Dessa forma, consegue-se identificar os objetos agrupá-los, de forma a encontrar suas classes conhecidas

13. VISÃO DE DADOS

Essa visão é utilizada em projetos onde existe alguma camada de duração, geralmente, um banco de dados racional. É uma visão geral dos dados persistentes, por meio dessa, objetos são mapeados dados persistentes. Essa visão é visualizada com o Modelo Entidade Relacionamento, no caso de banco de dados relacionais.

14. TAMANHO E DESEMPENHO

Descrição do desempenho e das características do software que impactam na arquitetura de software.

a. Requisitos Mínimos
   i. É necessário possuir conexão com a internet;
   ii. Para desenvolver, possuir: Windows, linux ou MacOS;

15. QUALIDADE

Qualidade de software tem como objetivo atingir os requisitos especificados durante a elaboração do projeto, e como expectativas de usuários de clientes e diretamente relacionados com: Escalabilidade, Manutenção, Confiabilidade, Usabilidade e assim por diante.