Diagrama

Descrição gerada automaticamente



Curso Superior de Desenvolvimento de Software Multiplataforma

Davy Oliveira Ribeiro - 3011392413004

Matheus Augusto Santos Gueff – 3011392413006

Pedro Silva Martins -

Ryan Carlo Negretti Pereira - 3011392413028

Projeto Interdisciplinar

Banco de Dados - Não Relacional

Desenvolvimento Web III

Gestão Ágil de Projetos de Software

Interação Humano Computador

Técnicas de Programação II

**SaneaSP**

**Orientadores**

Profa Esp. Cláudio Corredato

Profo Ms. Jones Artur Gonçalves

Profa Ma. Maria Janaína da Silva Ferreira

Prof. Ms. Ricardo Roberto Leme

Votorantim

**Versão 1.1** – Fev/2025

**Resumo**

O projeto SaneaSP tem como objetivo criar um site para fortalecer a comunicação entre a população e os responsáveis pela área de saneamento em Sorocaba/Votorantim, contribuindo para duas ODS: Água Potável e Saneamento (ODS6) e Saúde e Bem-Estar (ODS3). O projeto permitirá que os cidadãos relatem problemas de forma estruturada e fácil e acompanhem as soluções para esses problemas, além de disponibilizar informações organizadas sobre doenças relacionadas e ações corretivas e outras notícias relacionadas à área de saneamento básico. Dessa forma, buscamos contribuir para a eficiência e transparência das iniciativas já existentes.

**SUMÁRIO**

[1.](#_jevv97d8yno7) Tema Central 1

[1.1.](#_nqufzjf9usqj) Objetivo Geral do Projeto 1

[1.2.](#_elfxkwm569on) Implementações em cada disciplina 2

[[Disciplina-Chave] Gestão Ágil de Projetos de Software: 2](#_1yyg40jxqhg6)

[[Disciplina Satélite] Banco de Dados – Não Relacional 9](#_gwcs0gqyhpeh)

[[Disciplina Satélite] Interação Humano-Computador - IHC 11](#_ku6whqh6qa8k)

[[Disciplina Satélite] Desenvolvimento Web III 13](#_tdx64sknpv1s)

[[Disciplina Satélite] Técnicas de Programação II 14](#_4psw691lq351)

[Apresentação do Projeto no GitHub 14](#_8qurdgme0bv5)

[1.3.](#_7fb1hz9hck1) Formato da Entrega: 14

[2.](#_ince7t7h9zlo) REFERÊNCIAS 17

**LISTA DE FIGURAS**

Insira a lista de figuras.

**LISTA DE QUADROS**

Insira a lista de quadros.

# Tema Central

Desenvolvimento de uma solução Web (*back-end e front-end*) com integração com Banco de Dados Não Relacional.

## Objetivo Geral do Projeto

Criar uma aplicação web robusta e escalável, utilizando as melhores práticas de desenvolvimento, com as seguintes características:

**Desenvolvimento *Full-Stack*:** Construção completa da aplicação, abrangendo tanto a interface do usuário (*front-end*) quanto a lógica de negócio e acesso a dados (*back-end*).

**Tecnologias Modernas:** Emprego de linguagens de programação, *frameworks* e ferramentas de desenvolvimento web de última geração, garantindo alta performance e flexibilidade.

**Banco de Dados Não Relacional:** Adoção de um Sistema de Gerenciamento de Banco de Dados (SGBD) não relacional para armazenamento e gerenciamento eficiente de dados, otimizado para grandes volumes e alta escalabilidade.

**Integração REST:** Implementação de uma API RESTful para garantir uma comunicação eficiente e padronizada entre o front-end e o back-end. Essa abordagem permitirá a troca de dados de forma clara e segura, utilizando verbos HTTP (GET, POST, PUT, DELETE) e formatos de dados como JSON.

**Interface do Usuário Intuitiva e Acessível:** Desenvolvimento de uma interface do usuário (UI) que seja intuitiva, fácil de navegar e acessível a todos os usuários, independentemente de suas habilidades ou dispositivos. A UI deverá seguir as diretrizes de acessibilidade WCAG (*Web Content Accessibility Guidelines*) para garantir que a aplicação possa ser utilizada por pessoas com deficiência visual, auditiva, motora ou cognitiva.

**Controle de Versão:** Utilização de ferramentas de controle de versão para garantir a rastreabilidade das mudanças no código e facilitar a colaboração entre desenvolvedores.

## Implementações em cada disciplina

### [Disciplina-Chave] Gestão Ágil de Projetos de Software:

Neste projeto, você terá a oportunidade de vivenciar na prática os conceitos e práticas da **Gestão Ágil de Projetos de Software**, com foco na metodologia Scrum. O Scrum é uma abordagem ágil que divide o projeto em ciclos curtos chamados de ***sprints***, permitindo maior flexibilidade, adaptação e entrega contínua de valor.

**O que você vai aprender:**

* **Planejamento Ágil:** Dominar técnicas como a criação de *backlogs*, *user stories* e a priorização de funcionalidades.
* **Execução de *Sprints*:** Entender o funcionamento das cerimônias Scrum (planejamento, *daily, review* e retrospectiva) e como otimizar o trabalho em equipe.
* **Gerenciamento de Produtos:** Aprender a definir e gerenciar o *backlog* do produto, garantindo que o projeto esteja sempre alinhado com as necessidades do cliente.
* **Comunicação Eficaz:** Desenvolver habilidades de comunicação para facilitar a colaboração entre os membros da equipe e com o cliente.
* **Qualidade e Entrega Contínua:** Priorizar a qualidade do software e entregar valor ao cliente de forma incremental.

**O que você precisa entregar:**

* **Apresentações detalhadas:** Prepare apresentações que demonstrem o progresso do seu projeto a cada *sprint*, incluindo o *backlog*, as *user stories*, os critérios de aceitação e os resultados alcançados.
* **Documentação completa:** Mantenha uma documentação organizada do seu projeto, incluindo requisitos, arquitetura, diagramas, código-fonte e relatórios.
* **Protótipos funcionais:** Crie protótipos que demonstrem as funcionalidades do seu software e permitam coletar feedback do cliente.

O detalhamento sobre o que deve ser entregue, está definido a seguir:

**Entregas das *Sprints***

Ness tópico do projeto de desenvolvimento de software seguindo a metodologia Scrum, que é uma das abordagens mais populares no gerenciamento ágil de projetos. Cada equipe deverá preparar uma apresentação detalhada sobre as *sprints* desenvolvidas durante o projeto. Esta apresentação servirá tanto como uma entrega formal do trabalho realizado quanto como uma oportunidade para reflexão e aprendizado sobre o processo Scrum aplicado.

Explique as metas de cada *sprint* e os entregáveis específicos.

**Apresentação Final**

Neste tópico deve ser realizada uma Demonstração de Produto. Inclua vídeos ou capturas de tela do produto em funcionamento, se aplicável, para apoiar a demonstração prática.

***Backlogs & User Stories***

No desenvolvimento ágil de projetos usando Scrum, a definição e gerenciamento de *backlog* e *user stories* são elementos fundamentais. O *backlog* do produto é uma lista ordenada de tudo o que é necessário no produto, enquanto as *user stories* são descrições curtas e simples de uma funcionalidade contada da perspectiva do usuário final.

Para este exercício, vocês deverão apresentar o *backlog* e as *user stories* desenvolvidas para o seu projeto, explicando como cada item foi priorizado e detalhado ao longo do ciclo de desenvolvimento.

Cada equipe deverá preparar uma apresentação detalhada do *backlog* do produto e das *user stories* criadas durante o desenvolvimento do projeto. Esta apresentação deve demonstrar o entendimento da importância desses elementos no processo Scrum e a capacidade de criar descrições claras e úteis para o desenvolvimento ágil.

**Objetivos Específicos:**

Backlog do Produto:

Descrição Completa: Apresente o backlog do produto, incluindo todos os itens atualmente no backlog. Certifique-se de que a lista está ordenada por prioridade.

User Stories:

Criação de User Stories: Para cada item de maior prioridade no *backlog*, apresente as *user stories* correspondentes. Cada *user story* deve seguir o formato padrão (Como [tipo de usuário], eu quero [ação], para que [benefício]).

Critérios de Aceitação: Inclua critérios de aceitação claros para cada *user story*. Estes critérios ajudam a definir quando uma *user story* está completa e funcionando conforme esperado.

Ferramentas Utilizadas: Apresente as ferramentas utilizadas para gerenciar o *backlog* e as *user stories* (por exemplo, Trello, Jira, etc.). Mostre exemplos concretos do uso dessas ferramentas no projeto.

**Protótipo**

Prototipagem: Adicione documentos ou links para os protótipos desenvolvidos, descrevendo as ferramentas utilizadas para sua criação e explicando as funcionalidades principais demonstradas.

**Documentação do Projeto**

Documentação do Projeto: Inclua uma pasta /docs com todos os documentos relevantes, como o *backlog, user stories*, planejamento das *sprints*, revisões de *sprint*, lições aprendidas e reflexões.

O Levantamento de Requisitos, Requisitos Funcionais e Diagramas de Caso de Uso devem ser inclusos dentro desta documentação.

**Levantamento de Requisitos**

Nesse tópico, é necessário descrever a técnica que foi utilizada para levantamento dos requisitos. Por exemplo:

Questionários ou entrevistas com possíveis usuários

Análise do sistema antigo da empresa (caso exista uma empresa para a qual o software foi desenvolvido)

Pesquisa de mercado.

Se necessário, inclua documentos (coloque-os no Apêndice)

Também poderão ser descritas ferramentas existentes no mercado com funcionalidades semelhantes e que tenham sido utilizadas como base para a definição do projeto.

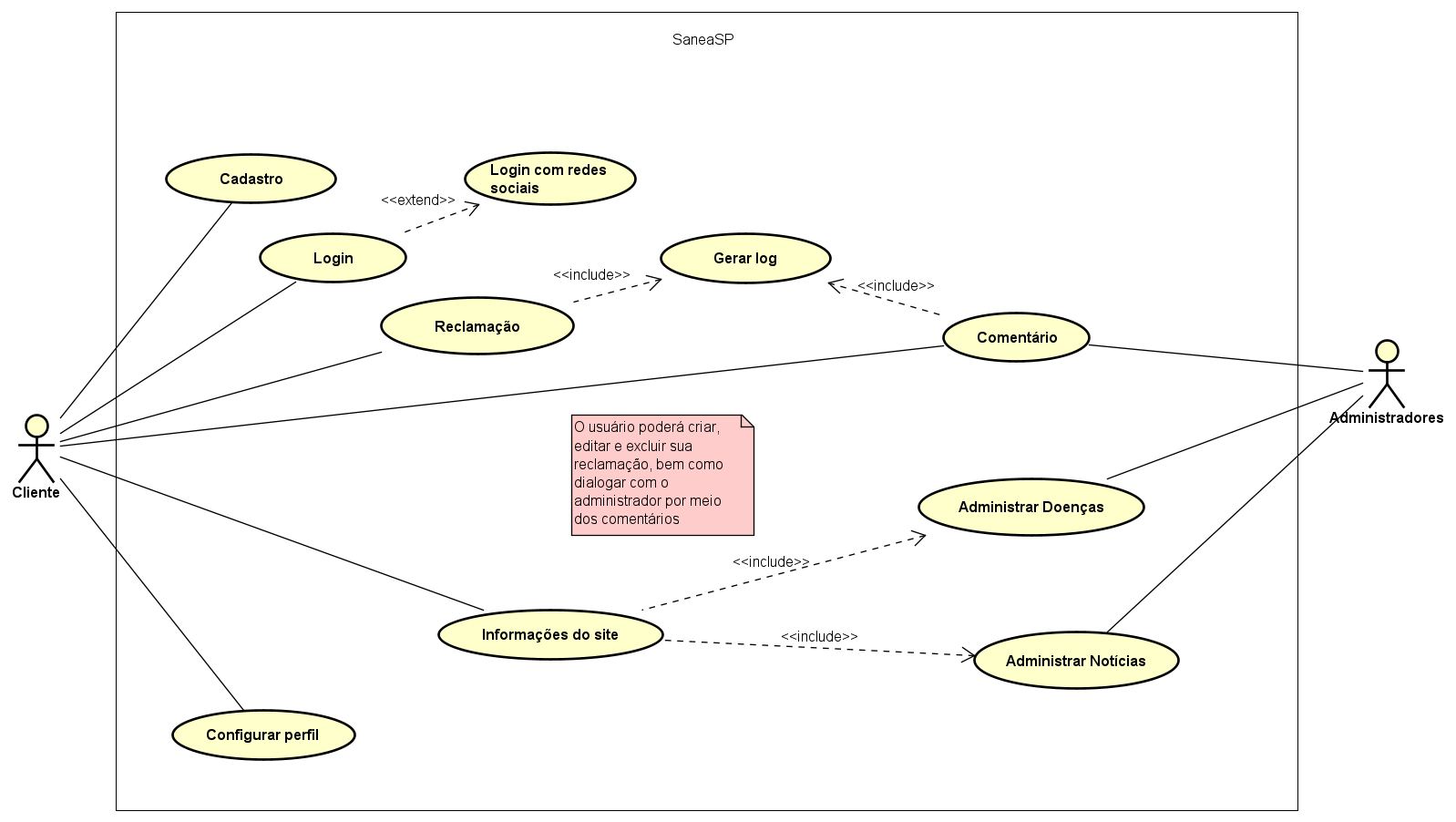
**Requisitos Funcionais**

|  |  |  |
| --- | --- | --- |
| Nº Requisito | Nome | Descrição |
| RF001 | Login e autenticação | O sistema deve permitir que os usuários façam login e autenticação pelo sistema ou com redes sociais para acessar funcionalidades conforme seu perfil. |
| RF002 | Gerenciamento e cadastro de reclamações | O usuário deve ser capaz de criar, editar e excluir reclamações, que serão públicas para visualização de outros usuários e administradores. |
| RF003 | Criação de Comentários e chat de respostas | O administrador pode responder reclamações e iniciar um chat para comunicação com o usuário que criou a reclamação. |
| RF004 | Gerenciamento e cadastro de doenças | O administrador pode criar, editar e excluir doenças relacionadas ao saneamento básico, que serão visíveis para todos os usuários |
| RF005 | Gerenciamento e cadastro de notícias | O administrador pode criar, editar e excluir notícias relacionadas a saúde pública e saneamento básico, que serão visíveis para todos os usuários |
| RF006 | Filtragem de reclamações | A tabela de reclamações deve ser filtrável por pontuação, categoria e localização geográfica. |
| RF007 | Upload de imagens | O sistema deve permitir o upload de imagens relacionadas aos registros para exibição no site. |
| RF008 | Logging de comentários e reclamações | Ações de criação, edição e exclusão de comentários e reclamações devem ser registradas em uma tabela de log. |
| RF009 | Geração de pontuação para reclamações | As reclamações devem ser classificadas por pontuação, baseada no detalhamento da reclamação e no perfil do usuário criador, para aumentar a confiabilidade. |

**Diagrama de Caso de Uso**

Insira seu diagrama de Casos de Uso.

Exemplo:



**Requisitos Não Funcionais**

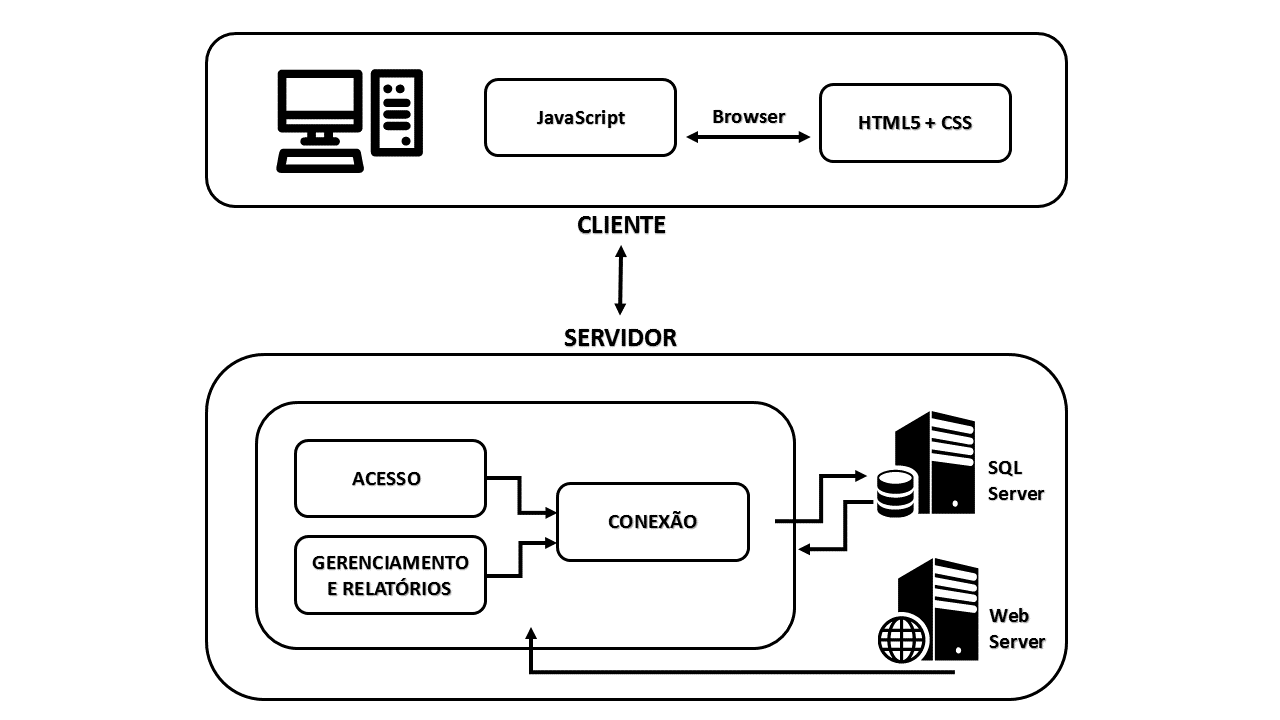
|  |  |  |
| --- | --- | --- |
| Nº Requisito | Nome | Descrição |
| RNF001 | Acesso à internet | O usuário deve dispor de conexão ativa à internet para acessar o site. |
| RNF002 | Banco de dados | O sistema deve utilizar banco de dados relacional para armazenamento e gerenciamento dos dados. |
| RNF003 | Responsividade | O site deverá ser responsivo, adaptando-se a diferentes resoluções de tela, com foco em uma boa experiência tanto em dispositivos móveis quanto em desktops. |
| RNF004 | Hardwares mínimos | O sistema poderá ser acessado por dispositivos com navegador moderno (Chrome 90+, Firefox 90+, Edge 90+, Safari 14+), e resolução mínima de 360x640. Recomendado uso em telas maiores (1366x768+) para administradores. |
| RNF005 | Hospedagem | O site deve estar hospedado em servidor que ofereça suporte a banco de dados e recursos web necessários para seu funcionamento. |
| RNF006 | Acessibilidade | O site deverá seguir diretrizes de acessibilidade (como WCAG), garantindo uso por pessoas com deficiência visual, incluindo cegos e daltônicos, além de oferecer recursos que melhorem a usabilidade para todos os usuários. |
| RNF007 | Feedback ao usuário | O sistema deve exibir notificações visuais ou sonoras para informar o usuário sobre erros, exceções ou a conclusão bem-sucedida de operações. |
| RNF008 | Backend e API | O sistema deve possuir backend robusto, expor API RESTful e integrar-se com banco de dados em nuvem, garantindo segurança e desempenho adequados. |
| RNF009 | Documentação | O projeto deve conter documentação atualizada, seguindo práticas das metodologias ágeis SCRUM e Kanban. |

**PROJETO DO SOFTWARE**

**Arquitetura da Aplicação**

Nesse tópico, apresentar de maneira sucinta qual foi o modelo arquitetural escolhido para o projeto. Além disso, é interessante incluir figuras facilitando o entendimento dos componentes, conforme exemplo apresentado na Figura 2.

Figura 2 - Arquitetura do Software



Fonte: Autoria própria

**Tecnologias Utilizadas**

Nesse tópico, é necessário descrever as tecnologias que serão utilizadas no desenvolvimento da aplicação, como por exemplo linguagem de programação, framework, banco de dados, API e hardware.

Linguagem de Programação: TypeScript

Frameworks: Angular, Bootstrap

Banco de dados: Utiliza o SQLite como banco de dados relacional, em conjunto com o Sequelize como ORM para facilitar a manipulação e modelagem dos dados.

API: A API REST da aplicação foi desenvolvida utilizando o framework Express.js, com as rotas organizadas para permitir operações CRUD sobre os recursos do sistema. A aplicação também consome dados da API pública do ViaCEP para busca de endereços por CEP.

**Integração e Papéis da Equipe**

Equipe: Crie uma lista com todos os integrantes do grupo e seus papéis no projeto. Especifique quem atuou como Scrum Master, Product Owner e membros do time de desenvolvimento. Inclua uma breve descrição das responsabilidades e contribuições de cada membro.

### [Disciplina Satélite] Banco de Dados – Não Relacional

O objetivo principal da disciplina Banco de Dados Não Relacional neste projeto é garantir que a solução web seja capaz de armazenar e gerenciar de forma eficiente grandes volumes de dados, com alta escalabilidade e flexibilidade.

Para alcançar esse objetivo, serão abordados os seguintes aspectos:

* **Modelagem de dados precisa e eficaz:** A modelagem de dados no banco não relacional será fundamental para representar de forma intuitiva e eficiente as coleções do sistema, como usuários, produtos, pedidos, etc. A escolha adequada de tipos de dados e a definição de relacionamentos entre documentos permitirão um armazenamento otimizado e consultas eficientes.
* **Implementação de consultas complexas:** A realização de consultas robustas e eficientes é essencial para extrair informações relevantes do banco de dados. Serão exploradas diferentes técnicas de consulta, como o uso de operadores de comparação, lógicos e projeções, além de junções entre coleções e o emprego do *aggregation pipeline* para análises mais complexas.
* **Otimização do desempenho:** A criação de índices adequados e a análise do desempenho das consultas são cruciais para garantir a rapidez e a escalabilidade da aplicação. Através do uso de ferramentas como o *explain* e o *db.collection.stats(),* será possível identificar gargalos e tomar medidas para otimizar o desempenho do banco de dados.
* **Suporte a análises de dados:** A capacidade de agrupar e agregar dados é fundamental para gerar *insights* valiosos sobre o negócio. Serão realizadas consultas de agregação para calcular estatísticas e gerar relatórios, permitindo uma melhor compreensão dos dados e a tomada de decisões mais assertivas.
* **Gerenciamento de dados geográficos:** Caso a aplicação exija o armazenamento de dados geográficos, a utilização do tipo GeoJSON permitirá realizar consultas espaciais e criar mapas interativos, enriquecendo a experiência do usuário.

Dessa maneira, a disciplina Banco de Dados Não Relacional neste projeto tem como objetivo garantir que o banco de dados seja uma peça fundamental para o sucesso da aplicação, proporcionando um armazenamento confiável, escalável e eficiente dos dados, além de permitir a realização de análises complexas e a geração de insights valiosos.

Para isso, será necessário a implementação dos seguintes itens:

* **Modelagem de Dados**:
  + Criar coleções para representar as diferentes entidades que irão compor o software (usuários, produtos, pedidos etc) e definir os esquemas adequados.
  + Utilizar tipos de dados como strings, números, datas, arrays e objetos aninhados.
  + Modelar relacionamentos entre documentos utilizando referências e arrays.
* **Consultas**:
  + Realizar no mínimo 5 consultas básicas utilizando operadores de comparação, lógicos e projeções.
  + Utilizar no mínimo 2 consultas com o operador $lookup para realizar junções entre coleções.
  + Empregar no mínimo em 1 consulta o aggregation pipeline para realizar análises complexas e transformar dados.
* **Índices**:
  + Criar índices para otimizar o desempenho de consultas frequentes.
  + Analisar o impacto dos índices no desempenho utilizando o explain e o db.collection.stats()
* **Agrupamento e Agregação**:
  + Utilizar em 2 consultas o $group para agrupar documentos por um campo específico.
  + Empregar em 2 consultas os operadores de agregação como $sum, $avg, $max, $min para calcular estatísticas.
* **Geospatial**:
  + Armazenar dados geográficos utilizando o tipo GeoJSON.
  + Realizar consultas geospaciais para encontrar documentos dentro de um raio específico ou dentro de um polígono.

### [Disciplina Satélite] Interação Humano-Computador - IHC

**Objetivos:** O aluno deverá aperfeiçoar a interface desenvolvida, implementando os requisitos da IHC, trabalhando os conceitos de usabilidade e acessibilidade.

**Requisitos do Projeto:**

1. Desenvolvimento da tarefa de Discovery, contendo:
   1. Técnica para identificação dos usuários (questionário, *brainstorm* etc)
   2. Definição das personas
   3. Definição dos cenários
2. Desenvolvimento da tarefa de arquitetura da informação:
   1. Estrutura da informação (*card sorting*)
   2. Menus e navegação
   3. Protótipo de baixa fidelidade
   4. Definição dos símbolos
   5. Definição das cores
   6. Definição dos Padrões
3. Prototipação, usabilidade e acessibilidade
   1. Protótipo de alta fidelidade
   2. Avaliação Heurística
   3. Desenvolvimento da interface WEB, utilizando os conceitos de usabilidade e acessibilidade adquiridos durante as aulas, conforme protótipo de alta fidelidade apresentado.

Para cada tópico será necessário elencar os objetivos, a proposta e o resultado obtido.

1. A entrega deverá ser feita através da tarefa disponibilizada na plataforma TEAMS. Onde um dos integrantes do grupo realizará a entrega de uma apresentação no Power Point contendo os itens elencados anteriormente. Certifique-se de incluir o nome de todos os integrantes do grupo na apresentação e incluir o link PÚBLICO do FIGMA.

**Entregas:**

A primeira entrega consiste na apresentação da pesquisa realizada pelos integrantes do grupo. Serão avaliados na entrega, os itens 1 e 2.

A segunda entrega consiste na apresentação do protótipo pelos integrantes do grupo e entrega da apresentação na tarefa do Teams. Serão avaliados na entrega, os itens de 1 até 3.

[Disciplina Satélite] Desenvolvimento Web III

**Objetivo**: Desenvolvimento de uma API REST, na arquitetura MVC, utilizando o banco de dados MongoDB e integração com o Front-End.

**Requisitos do Projeto:**

* 1. A aplicação deverá ser hospedada no GitHub e ter sua documentação descrita no arquivo README. Não esqueça de incluir o nome dos integrantes do grupo;
  2. Desenvolvimento uma *API RESTful* completa que permita a realização das operações básicas: GET, POST, PUT e DELETE. Cada operação deve ser mapeada para as rotas apropriadas no seu servidor;
  3. Utilização da arquitetura MVC para desenvolvimento da aplicação;
  4. Deverá conter obrigatoriamente um microsserviço;
  5. Aplicação hospedada em nuvem, através de uma plataforma, por exemplo: Vercel
  6. API documentada utilizando uma das ferramentas de documentação apresentadas em aula. Exemplo Postman ou Swagger;
  7. Implementação de um sistema de login, com autenticação dos usuários.
  8. Implementação de um sistema de proteção das rotas, utilizando autenticação através de token, api key ou outros;
  9. Desenvolvimento da interface do usuário utilizando o conceito de SPA (*Single Page Application*), poderá ser utilizado o *framework* de preferência do grupo (Angular, React etc);
  10. Realização dos testes unitários, através do *framework* Jest.
  11. A entrega deverá ser feita através da tarefa disponibilizada na plataforma TEAMS. Onde um dos integrantes do grupo realizará a entrega do link do repositório PÚBLICO do GitHub. Certifique-se de incluir o nome de todos os integrantes do grupo no README do projeto e o link da aplicação para facilitar a identificação dos colaboradores, assim como o link da API pública, se houver.

**Entregas**:

A primeira entrega consiste na apresentação da API desenvolvida pelos integrantes do grupo. Serão avaliados na entrega, os itens de 1 até 5. A segunda entrega consiste na apresentação do projeto pelos integrantes do grupo e entrega do link do repositório na tarefa do Teams. Serão avaliados na entrega, os itens de 1 até 10.

[Disciplina Satélite] Técnicas de Programação II

**Objetivo**: Técnicas de Programação II.

Objetivos: Os discentes deverão aperfeiçoar as interfaces de administração de seus respectivos projetos integradores, adicionando novas funcionalidades e comportamentos. Haverá acesso ao banco de dados nessa etapa.

Primeira Entrega: Documento descrevendo as modificações e aperfeiçoamentos que serão implementados. Diagrama UML de classes, incluindo as classes de acesso ao banco de dados.

Segunda Entrega: Descrição detalhada dos testes de validação, utilizando de forma mais profunda os conceitos de TDD.

Terceira Entrega: Classes de acesso ao banco de dados e apresentação das novas funcionalidades, acompanhado dos testes de validação e modelos dos bancos de dados.

Quarta Entrega: Integração das novas classes ao projeto, acompanhado dos testes de validação.

Quinta Entrega: Apresentação do projeto funcional e entrega da documentação atualizada.

**Requisitos do Projeto:**

Apresentação do Projeto no GitHub

A gestão de código e documentação em um repositório centralizado é uma prática essencial em projetos de desenvolvimento de software. GitHub é uma das plataformas mais utilizadas para este propósito, facilitando a colaboração, controle de versão, e revisão de código. Nesta etapa final, vocês irão consolidar todo o trabalho realizado ao longo do projeto em um repositório no GitHub.

Cada equipe deverá criar um repositório no GitHub para o projeto de desenvolvimento de software. Este repositório deve incluir todos os elementos desenvolvidos no projeto, como o *backlog* do produto*, user stories*, planejamento das *sprints*, e demonstração do produto. A documentação deve ser clara, organizada e refletir todo o trabalho realizado ao longo do ciclo de desenvolvimento.

## Formato da Entrega:

**Repositório GitHub**: O repositório deve estar completo e acessível publicamente até a data de entrega.

**Documentação Completa**: Todas as documentações, incluindo *backlog, user stories*, planejamento das *sprints*, revisões de *sprint*, código-fonte, demonstrações, tecnologias utilizadas, e reflexões, devem estar incluídas e organizadas.

**Apresentação Final**: Prepare uma breve apresentação para demonstrar o repositório GitHub, destacando como cada parte do trabalho foi organizada e documentada.

**Exemplo de um projeto:** Segue o link com um exemplo de um projeto criado por uma outra turma, que pode servir como um modelo para o projeto.   
<https://github.com/The-Bugger-Ducks/help-duck-documentation>

# REFERÊNCIAS

Listar somente as referências que têm autoria e que foram efetivamente citadas no texto.

As referências sem autoria, representadas apenas por uma URL (Ex. http://pmkb.com.br/sig/padroes-frameworks/pmbok-pmi/) devem ser apresentadas ao longo do texto, em notas de rodapé, de acordo com o exemplo a seguir:

¹ Conforme disponível em: < http://pmkb.com.br/sig/padroes-frameworks/pmbok-pmi/>. Acesso em: 10 jul. 2020.

Exemplo de referência:

SOMMERVILLE, Ian. **Engenharia de Software**. 8ª ed. Pearson, 2007

Para a correta geração das referências, sugere-se utilizer o site da UFSC, chamado MORE – Mecanismo Online de Referências: <https://more.ufsc.br/>

**Anexos**

Esse tópico é opcional, no qual podem ser inseridos documentos agregados à obra para fins de comprovação de dados ou ilustração.

**Apêndice**

Esse tópico é opcional, no qual podem ser inseridos documentos de agregados à obra para fins de apoio à argumentação. Nesta parte, são inclusos os questionários, entrevistas, tabulação de dados, entre outros.