Projeto Integrador 3º Semestre - DSM

**Disciplinas:**

Gestão Ágil de Projeto de Software

Banco de Dados Não Relacional

Desenvolvimento Web III

**Professores:**

Renato Cividini Matthiesen

Thiago Gonçalves Mendes

Orlando Saraiva do Nascimento Junior

**Grupo(n) / Nome da Empresa:**

**Sistema:**

|  |  |
| --- | --- |
| **Integrantes** | **Papel Principal** |
| Jonathan de Santana Rocha | Back-End |
| Kalliel Pinheiro | Back-End |
| Luiz Henrique Simionato Vicente | Front-End / Design |
| Marco Antonio Amorim Filho | Front-End |
|  |  |

**Fatec Araras**

**2024**

FICHA DE CONTROLE - PROJETO INTERDISCIPLINAR

DISCIPLINA CHAVE: Gestão Ágil de Projeto de Software

PROFESSOR: Renato Cividini Matthiesen

GRUPO: 3 SEMESTRE: 3

TÍTULO DO PROJETO: InovaDev Consultoria

DATA DA APRESENTAÇÃO: 07/12/2024

NOTA:

INTEGRANTES DO GRUPO:

|  |  |
| --- | --- |
| Nome | Nota Individual |
| Jonathan de Santana Rocha |  |
| Kalliel Pinheiro |  |
| Luiz Henrique Simionato Vicente |  |
| Marco Antonio Amorim Filho |  |
|  |  |
|  |  |
|  |  |
|  |  |

Araras,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor Renato Cividini Matthiesen

* Gestão Ágil de Projeto de Software

Fazer uma breve revisão bibliográfica e considerações sobre a abordagem ágil de projeto de software. Verifique as sugestões de uso de bibliografias em Referências Bibliográficas.

. Documentação oficial Django

. Documentação oficial Python

. Metodologia Scrum

. Documentação oficial MongoDB

**1.1 Apresentação do projeto e considerações sobre a empresa (fictícia) que está desenvolvendo o sistema**

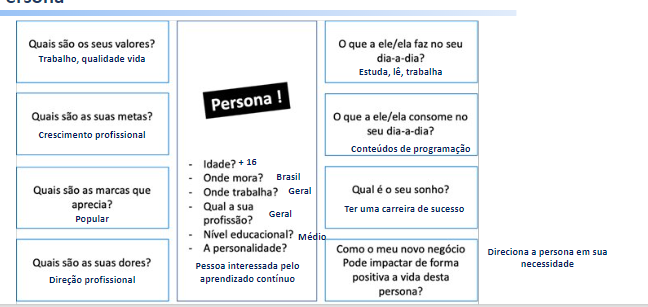
**.** A InovaDev Consultoria é uma startup formada por quatro alunos dedicados: Jonathan, Kalliel, Luiz e Marcos. Somos uma equipe que acredita no poder do compartilhamento de conhecimento para acelerar o crescimento profissional. Nossa missão é conectar pessoas que buscam orientação com mentores experientes, utilizando metodologias ágeis para garantir a entrega de um produto de alta qualidade.

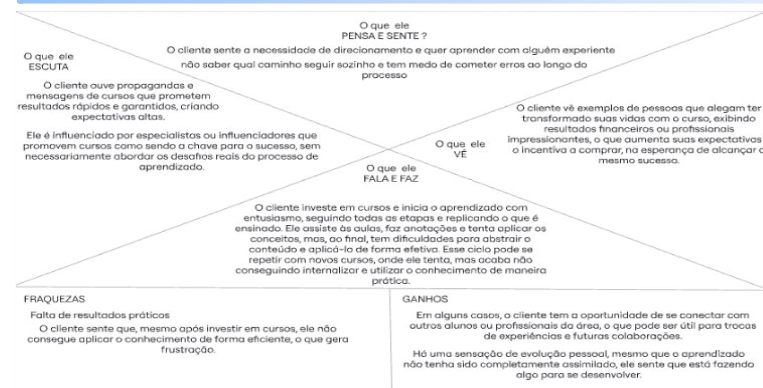
**1.2 Visão do produto**

**.** Plataforma web de mentoria individual para programadores, tanto iniciantes quanto experientes, que conecta os usuários a especialistas da área por meio de videochamadas. A plataforma promove o aprendizado contínuo dentro de uma comunidade colaborativa.​

**1.3 Persona + Mapa de Empatia**

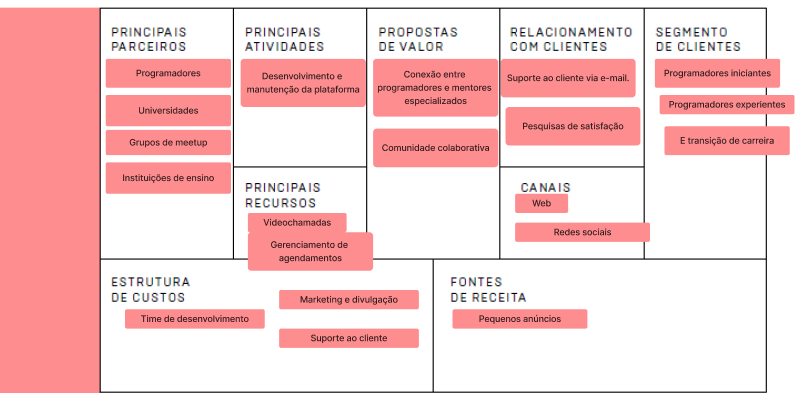
Elaborar a persona e/ou mapa de empatia para o perfil de clientes/usuários.





**1.4 Canvas do projeto/produto**

Elaborar o Business Model Canvas do projeto.



**1.5 Mapa mental do sistema**



**1.6 *Backlog* do Produto**

Requisitos Funcionais:

**Cadastro de Usuários**

Como usuário, quero me cadastrar na plataforma usando CPF e senha para acessar os recursos.

. Critérios de Aceitação:

. Formulário de cadastro com validação de CPF.

**Login e Autenticação**

Como usuário, quero fazer login para acessar minha conta com segurança.

. Critérios de Aceitação:

. Sistema deve permitir login usando CPF e senha.

**Criação de Perfil de Mentor**

Como mentor, quero criar e editar meu perfil para exibir minhas qualificações e áreas de especialidade.

. Critérios de Aceitação:

. Campos para nome, experiência, especialidades e links de redes sociais.

. Opção de atualizar o perfil a qualquer momento.

**Busca de Mentores**

Como usuário, quero pesquisar mentores por especialidade para encontrar orientação adequada.

. Critérios de Aceitação:

. Filtro de busca por especialidade.

. Exibir lista de mentores com informações resumidas.

**Agendamento de Sessões de Mentoria**

Como mentorado, quero agendar sessões de mentoria para receber orientação personalizada.

. Critérios de Aceitação:

. Sistema de escolha de data e horário disponível.

**Chat ao Vivo**

Como usuário, quero ter um chat ao vivo para facilitar a comunicação com o mentor durante as sessões.

.Critérios de Aceitação:

.Chat em tempo real integrado à sessão.

Requisitos Não Funcionais:

**Segurança de Dados**

Como administrador, quero que todos os dados dos usuários sejam armazenados de forma segura para garantir a privacidade.

.Critérios de Aceitação:

.Criptografia dos dados sensíveis (senhas e informações pessoais).

. Conformidade com normas de segurança e privacidade.

**Performance e Escalabilidade**

Como usuário, quero que a plataforma responda rapidamente para que minha experiência seja fluida e eficiente.

. Critérios de Aceitação:

. Tempo de resposta das páginas abaixo de 3 segundos.

. Suporte a vários usuários simultâneos sem perda de desempenho.

**Backup e Recuperação de Dados**

Como administrador, quero ter backups diários para evitar perda de dados em caso de falhas.

**Conformidade com LGPD**

Como usuário, quero ter controle sobre meus dados pessoais para garantir a conformidade vigente.

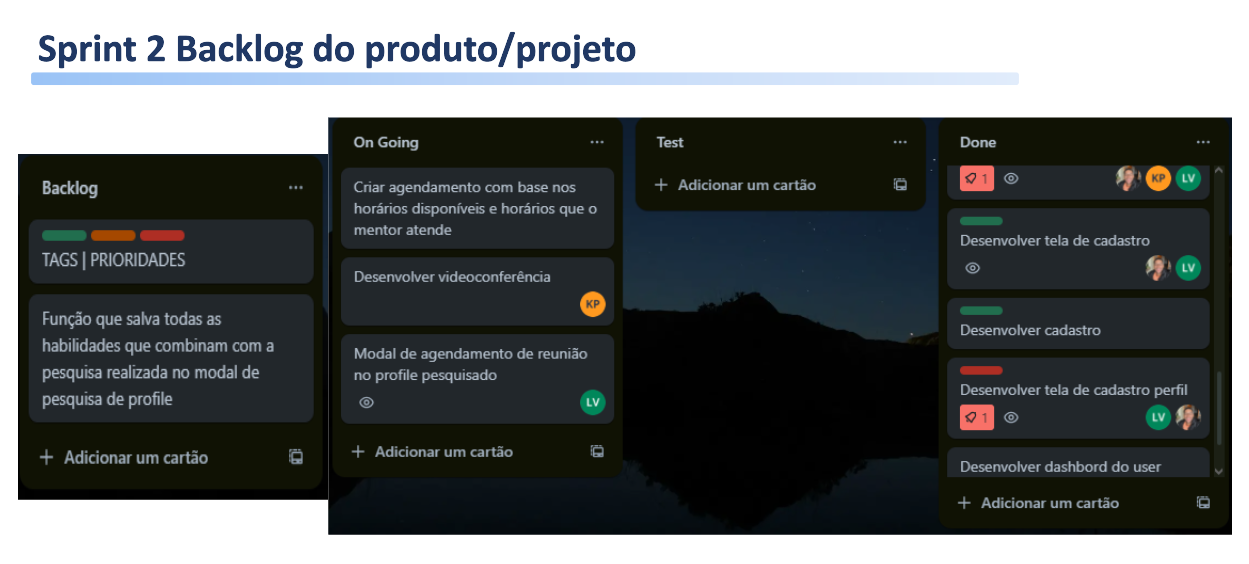
. Critérios de Aceitação:

. Opção de visualizar e editar dados pessoais.

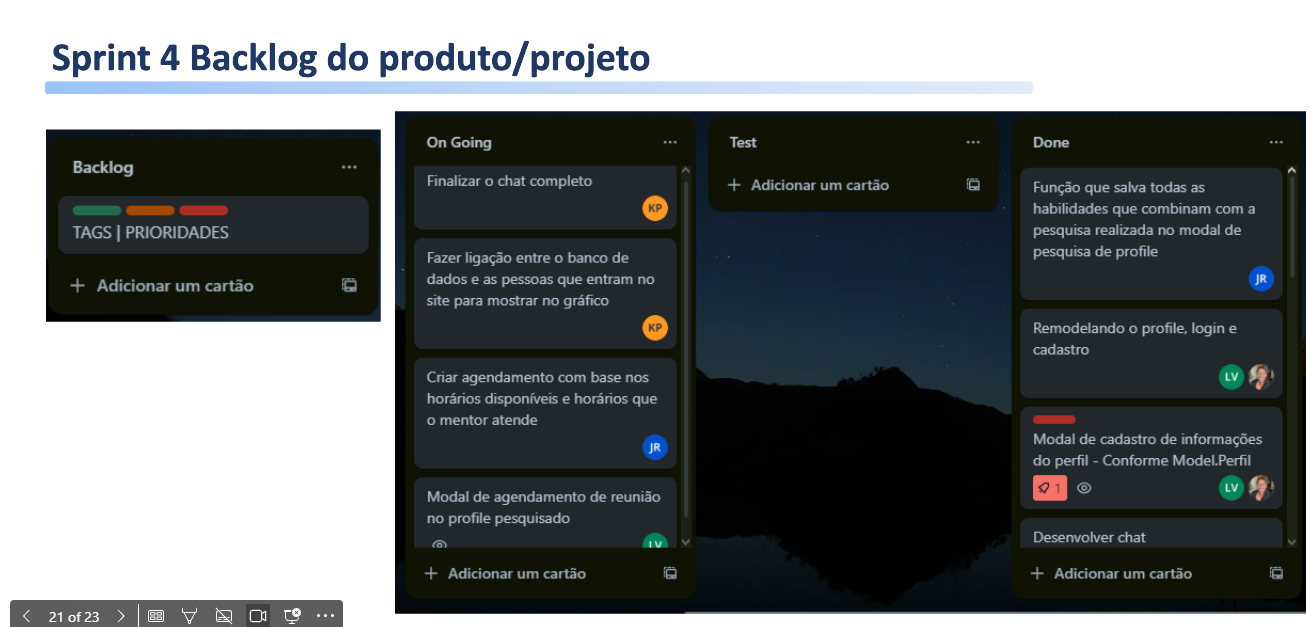
. Política de privacidade clara e acessível.

**1.7 *Backlog* dos *Sprints***









**1.8 Entregáveis – *Minimum Viable Product* (MVP)**

Apresentar as diferentes versões dos MVPs do produto em formato de entregáveis realizados durante as seis semanas de atividades do projeto.

Sugestão de bibliografia:

MATSUMOTO, Mayra, **MVP**: a forma eficiente de criar, validar e comercializar um novo produto ou serviço, Kindle: Amazon.com, 2020.

RIES, E. **A Startup Enxuta.** São Paulo: Editora Leya, 2019.

SUTHERLAND, J. **Scrum**: a arte de fazer o dobro do trabalho na metade do tempo. Rio de Janeiro: Sextante, 2019.

SCRUMSTUDY, **Guia SBOK: Guia para o Conhecimento em Scrum**, Arizona, EUA: SCRUMstudy, 2016. [https://www.Scrumstudy.com/SBOK/SCRUMstudy-SBOK-Guide-2016-Portuguese.pdf](https://www.scrumstudy.com/SBOK/SCRUMstudy-SBOK-Guide-2016-Portuguese.pdf). Acesso em: 08 mar. 2024.

**1.9 Retrospectiva dos *Sprints***

1. Pontos Positivos:

Comunicação Eficiente: A equipe demonstrou uma comunicação clara e contínua, especialmente nas Daily Scrum. Isso facilitou o alinhamento das tarefas e a resolução rápida de problemas.

Entrega de Incrementos Valiosos: Concluímos todas as funcionalidades planejadas dentro do prazo, incluindo o cadastro de mentores e a busca por mentores.

Adaptação ao Feedback: A equipe soube adaptar o backlog de produto rapidamente após receber feedback dos usuários, o que melhorou a experiência inicial na plataforma.

2. Oportunidades de Melhoria:

. Estimativa de Tarefas: Em algumas Sprints, a equipe subestimou o tempo necessário para completar certas tarefas, o que causou uma sobrecarga de trabalho e atrasos na entrega de funcionalidades.

. Gerenciamento de Impedimentos: Identificamos que alguns obstáculos não foram resolvidos rapidamente devido à falta de uma abordagem proativa, o que impactou negativamente a produtividade da equipe.

. Reuniões Longas: Algumas reuniões de planejamento e revisão foram mais longas do que o esperado, o que prejudicou a eficiência e a concentração da equipe.

3. Ações de Melhoria para Próximas Sprints:

. Aprimoramento no Gerenciamento de Impedimentos: O Scrum Master irá monitorar de perto os impedimentos e implementar um sistema de priorização para garantir que os problemas sejam resolvidos rapidamente.

. Reuniões Mais Curtas e Objetivas: O time se comprometeu a seguir uma pauta mais estruturada para as reuniões de planejamento e revisão, visando manter o foco e reduzir o tempo gasto.

**1.10 Retrospectiva do Projeto.**

1. Pontos Fortes:

Boa aplicação do Scrum: A metodologia ajudou a manter o projeto organizado e adaptável.

Trabalho em Equipe: Comunicação eficiente e colaboração entre os integrantes Jonathan, Kalliel, Luiz Henrique e Marco

Entrega de Valor: O produto final atendeu às expectativas, facilitando a conexão entre mentores e mentorados.

2. Desafios Encontrados:

Estimativas Imprecisas: Algumas tarefas foram subestimadas, resultando em atrasos.

Resolução de Impedimentos: Problemas técnicos demoraram mais do que o previsto para serem resolvidos.

3. Ações para Melhorar:

Implementar uma análise mais cuidadosa para mudanças de escopo.

Melhorar o processo de gestão de impedimentos.

* **Desenvolvimento Web III**
* [**https://github.com/jonathannrocha/FATEC\_2024\_3SEM\_PI/graphs/contributors**](https://github.com/jonathannrocha/FATEC_2024_3SEM_PI/graphs/contributors)
* **Banco de dados Não Relacional**

**O uso de bancos de dados não relacionais, ou NoSQL, tem sido cada vez mais valorizado devido ao crescimento exponencial no volume de dados e à maior complexidade dos mesmos. Nos bancos de dados relacionais tradicionais, os dados são organizados em tabelas e um esquema rígido é necessário, nos bancos NoSQL é permitida uma estrutura flexível, adaptando-se melhor às demandas de dados variados e dinâmicos. De acordo com Boaglio (2015), a flexibilidade dos esquemas de bancos NoSQL, como o MongoDB, é destacada, permitindo que dados sejam inseridos sem a necessidade de um esquema pré-definido e facilitando mudanças rápidas nas aplicações.**

**A escalabilidade horizontal também é oferecida pelos bancos de dados NoSQL, o que torna possível o armazenamento de grandes volumes de dados em várias máquinas, sem a perda de desempenho. Esse recurso é enfatizado por Sadlage e Fowler (2013), que explicam como essa escalabilidade permite que clusters de servidores comuns sejam utilizados para atender a grandes quantidades de dados, mantendo a eficiência no acesso. Em sistemas de grande escala, como redes sociais e plataformas de streaming, essa capacidade é considerada essencial para garantir alta disponibilidade e performance frente a volumes elevados de transações.**

**Além disso, modelos de dados distintos são suportados por bancos NoSQL, entre eles documentos, chave-valor, colunares e grafos, o que facilita o uso em aplicações que requerem estruturas de dados variadas. Segundo Elmasri e Navathe (2019), os bancos NoSQL são especialmente indicados para dados semi-estruturados ou não estruturados, como texto e imagens, e para redes de conexões complexas, contrastando com os bancos de dados relacionais, que atendem melhor a dados altamente estruturados. Assim, uma maior variedade de dados pode ser suportada e tratada com eficiência.**

**A persistência poliglota, prática pela qual diferentes tipos de bancos de dados são combinados para atender necessidades específicas de uma aplicação, também é facilitada pelos bancos de dados NoSQL. Esse conceito é explorado por Singh (2001), que ressalta a importância de se escolher o modelo de dados mais adequado para cada contexto. Em sistemas híbridos, bancos de dados NoSQL são utilizados para armazenar dados que demandam rápida leitura e escrita, enquanto os bancos relacionais ficam responsáveis por dados com relações mais complexas, criando um ambiente eficiente e adaptável**.

No 3º semestre, enfrentamos o desafio de integrar o MongoDB ao Django. Como o Django 5.1.3 não oferece suporte nativo ao MongoDB, tivemos que buscar alternativas. Dentro do contexto da nossa aplicação, o MongoDB se encaixa bem para armazenar perfis de usuários e as salas de bate-papo (rooms). Esses dados são frequentemente flexíveis e instáveis, e o MongoDB facilita o armazenamento e o crescimento exponencial das mensagens trocadas nas salas. Além disso, o fluxo de dados para insights dos usuários pode ser gerido de forma mais eficiente no MongoDB, aproveitando sua escalabilidade horizontal.

Entretanto, surge a dúvida: por que não centralizar tudo no MySQL? A resposta está em aproveitar as vantagens dos dois tipos de banco de dados, cada um no papel que desempenha melhor dentro da aplicação.

De um lado, temos o PostgreSQL, que é ideal para armazenar dados sensíveis dos usuários e para o controle de autenticação, já que a aplicação exige um alto nível de integridade e segurança para essas informações. A utilização de chaves primárias (PK) e estrangeiras (FK) no modelo relacional garante que a integridade referencial entre as entidades seja mantida, o que é crucial para o controle de usuários e suas permissões dentro do sistema.

Por outro lado, o MongoDB oferece flexibilidade ao permitir o armazenamento de dados menos estruturados, como as mensagens e outros dados temporários. Utilizando o Mongo Engine com o Django ORM, conseguimos aproveitar a escalabilidade horizontal do MongoDB para gerenciar o crescimento das salas de bate-papo, sem a necessidade de um esquema rígido. Isso facilita a adição de novos recursos no futuro, sem a preocupação com dependências entre as entidades e mantendo a performance e agilidade da aplicação, principalmente no que diz respeito ao gerenciamento de mensagens e insights dos usuários

* **CRUD E ETL**

**CRUD (Create, Read, Update, Delete) e ETL (Extract, Transform, Load) são conceitos fundamentais no gerenciamento e manipulação de dados, mas desempenham papéis diferentes em sistemas de banco de dados. Enquanto o CRUD define as operações básicas para a manutenção de registros em um banco de dados, o ETL é focado na preparação e movimentação de dados entre sistemas, geralmente com o objetivo de análise e tomada de decisão (Elmasri & Navathe, 2019).**

**No contexto de um banco de dados transacional, como um sistema de vendas, as operações CRUD permitem que os dados sejam manipulados em tempo real: o usuário pode criar um novo registro (por exemplo, adicionando uma nova venda), ler ou visualizar dados (como consultar um histórico de vendas), atualizar informações (alterando o valor de uma venda), ou deletar registros. Esse modelo é ideal para aplicativos que exigem interações constantes com o banco de dados, garantindo que as informações estejam sempre atualizadas para atender a atividades operacionais (Boaglio, 2015).**

**Por outro lado, o ETL se concentra em extrair dados de sistemas transacionais ou outras fontes, transformá-los em um formato que atenda às necessidades analíticas e carregá-los em um data warehouse ou banco de dados analítico. Em um cenário de análise de vendas, por exemplo, o ETL seria responsável por consolidar os dados das vendas de diferentes sistemas em um único local, aplicando transformações para padronizar informações como formatos de data ou moeda e garantindo que o conjunto de dados seja coeso e pronto para análise (Singh, 2001). Dessa forma, o ETL apoia a criação de relatórios, dashboards e análises preditivas, com dados historicamente consistentes e atualizados periodicamente.**

**Ambos os processos podem ser complementares em um ambiente de data warehousing. As operações CRUD mantêm o sistema transacional atualizado, enquanto o ETL periodicamente extrai esses dados para o ambiente analítico. Essa integração é essencial para permitir que as organizações não apenas executem suas operações diárias com precisão, mas também utilizem os dados operacionais de forma a gerar insights estratégicos (Sadalage & Fowler, 2013). Portanto, CRUD e ETL formam uma base sólida para a gestão de dados em múltiplos níveis, com o CRUD mantendo a integridade operacional e o ETL apoiando a análise e o planejamento estratégico.**

Dentro do contexto de salas de bate-papo, perfis de usuários e armazenamento de dados de pesquisa, o NoSQL se destaca significativamente, especialmente em termos de processamento e serialização de dados. Isso ocorre porque o NoSQL, com sua estrutura menos rígida, oferece uma flexibilidade interessante que pode ser facilmente explorada na nossa aplicação. Ao utilizar um esquema semiestruturado, conseguimos adicionar novas funcionalidades sem comprometer a estrutura ou a funcionalidade existente. Essa flexibilidade é um dos principais pontos positivos de um banco NoSQL, permitindo um crescimento mais ágil e sem as limitações de um esquema tradicional.

Em termos de desempenho, para o nosso caso específico, em que não usamos funcionalidades complexas como agregações ou Joins, o NoSQL apresenta um desempenho muito superior. Isso se deve à sua natureza de consulta direta nas coleções de dados, sem a necessidade de verificar relacionamentos entre tabelas. Por exemplo, se simulássemos uma consulta em um banco de dados com 100 mil usuários (ou até volumes de dados muito maiores), o NoSQL concluiria a operação muito mais rapidamente. Isso porque o banco de dados NoSQL simplesmente filtra a query e retorna os dados desejados, sem a sobrecarga de verificar tabelas associadas, como ocorre no MySQL. No MySQL, a consulta envolve uma série de verificações de relações e dependências entre tabelas, o que pode tornar a operação significativamente mais lenta, especialmente em grandes volumes de dados

* **A Importância de Dashboards e Consultas Estratégicas para a Tomada de Decisão Empresarial**

Dashboards e consultas de valor para o negócio são ferramentas essenciais para transformar dados brutos em informações acionáveis, oferecendo uma visão clara e concisa do desempenho da organização. Os dashboards, muitas vezes definidos como painéis de controle visual, consolidam indicadores-chave de desempenho (KPIs) e métricas importantes em uma interface interativa, permitindo que gestores e analistas acompanhem as operações e tomem decisões informadas em tempo real (Singh, 2001). Esse recurso visual é crucial para simplificar dados complexos e torná-los acessíveis, de modo que mesmo os tomadores de decisão que não possuem um conhecimento técnico profundo possam entender rapidamente as principais tendências e desafios enfrentados pela organização.

Consultas de valor para o negócio, por sua vez, envolvem a realização de análises orientadas para responder questões específicas e relevantes para os objetivos estratégicos da organização. Essas consultas são voltadas para a extração de insights que apoiam decisões sobre, por exemplo, a identificação de padrões de comportamento do cliente, o monitoramento de despesas e receitas, ou a análise de desempenho de produtos e serviços. De acordo com Elmasri e Navathe (2019), diferente de consultas operacionais simples, que recuperam dados específicos para tarefas de rotina, as consultas de valor têm uma profundidade analítica maior, explorando os dados para responder a perguntas críticas para o sucesso do negócio e identificando áreas de oportunidade e melhoria.

Ao serem integrados a dashboards, os resultados das consultas de valor ampliam ainda mais a capacidade analítica do sistema, pois permitem que as informações derivadas dessas consultas sejam visualizadas de forma dinâmica e visual. Isso facilita a compreensão dos dados e possibilita uma análise comparativa e temporal, ajudando os gestores a identificar rapidamente mudanças e tendências (Sadalage & Fowler, 2013). Além disso, a atualização em tempo real dos dashboards proporciona uma visão atualizada, que é essencial para empresas que operam em mercados dinâmicos e competitivos. Quando bem projetados, esses dashboards vão além da exibição de dados e promovem a exploração de insights, permitindo a combinação de métricas e a geração de relatórios detalhados.

A importância dos dashboards e das consultas de valor para o negócio se reflete na maneira como essas ferramentas impactam diretamente a tomada de decisão. Dados isolados dificilmente têm valor estratégico; é a análise e a interpretação por meio de ferramentas visuais e consultas relevantes que conferem significado e contexto a esses dados, transformando-os em ativos estratégicos (Boaglio, 2015). Essa abordagem permite que decisões sejam embasadas em dados concretos, aumentando a precisão e a agilidade nas respostas às demandas do mercado. Em última instância, dashboards e consultas de valor para o negócio tornam-se um diferencial competitivo, capacitando as organizações a serem mais proativas e a responderem de forma eficaz aos desafios e oportunidades do ambiente corporativo.

A plataforma busca fornecer dados em tempo real sobre as tendências de aprendizado de linguagens de programação, permitindo que os usuários vejam o que está sendo mais buscado por outros. O NoSQL, com sua estrutura flexível e capacidade de lidar com dados não estruturados ou semiestruturados, é ideal para armazenar esse tipo de informação. Além disso, a facilidade em adicionar novos tipos de dados sem precisar alterar a estrutura do banco é uma vantagem clara do NoSQL. Caso decidamos adicionar novas funcionalidades, como recomendações personalizadas de linguagens com base no histórico de aprendizado do usuário, podemos fazer isso rapidamente, sem precisar reformular a arquitetura do banco de dados

A construção do nosso projeto está diretamente relacionada às características do banco de dados NoSQL, que, devido à sua flexibilidade e escalabilidade, se encaixa perfeitamente nas necessidades específicas da plataforma. Em nossa plataforma, exibimos quais linguagens de programação estão sendo mais buscadas por outros usuários, oferecendo um direcionamento para aqueles que buscam aprimorar suas habilidades e alcançar um nível superior no aprendizado de programação. Esse tipo de dado, que reflete o comportamento real e as escolhas de outros usuários, exige uma abordagem dinâmica e de fácil adaptação – exatamente o que o NoSQL oferece.

Kalliel:

**.** Back-end, Django, MongoDB, PostgreSQL e WebSockets. No Django, consegui explorar diversos pontos, tanto nativos quanto não nativos, como o fluxo de dados no MongoDB e a integração com WebSockets, que não possuem suporte nativo. Além disso, utilizei de forma aprofundada as principais classes, como Django.Core, django.http, django.db, django.urls, django.template, django.forms, django.contrib, django.middleware, django.conf e django.apps, para otimizar a estrutura e a funcionalidade da aplicação.

Luiz Henrique:

Como desenvolvedor Front-end do projeto, fui responsável por transformar o design da equipe em uma aplicação funcional, interativa e responsiva, sempre com foco em oferecer a melhor experiência para o usuário. Trabalhei na criação da estrutura visual da plataforma utilizando HTML, CSS e JavaScript, desenvolvendo as páginas principais, como a de login e cadastro, dashboard, perfil do mentor, busca de mentores e agendamento de sessões de mentoria. Desde o início, me preocupei em garantir um layout responsivo, utilizando técnicas de CSS Flexbox e Grid, além de frameworks como Bootstrap, para que o site fosse acessível e oferecesse uma boa experiência. Na parte do banco de dados consegui explorar diversos pontos, tanto nativos quanto não nativos, como o fluxo de dados no MongoDB

Jonathan:

Com o projeto foi possível implementar diversos conhecimentos adquiridos ao longo do semestre como: Django, gestão de projetos e banco de dados não relacional. Tive como escopo no projeto, atuar principalmente com back-end, implementando funcionalidades. Meu principal aprendizado e primeiro contato, foi como o Django, uma ferramenta simples e que agiliza o processo de desenvolvimento, com diversas abstrações e ferramentas. Mas nesse meio tempo podemos também observar suas limitações justamente com a ferramenta de banco de dados não relacional o Mongodb.

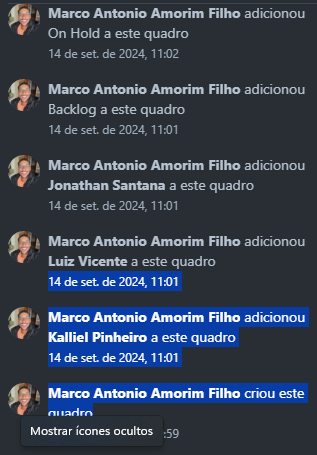
Marcos:

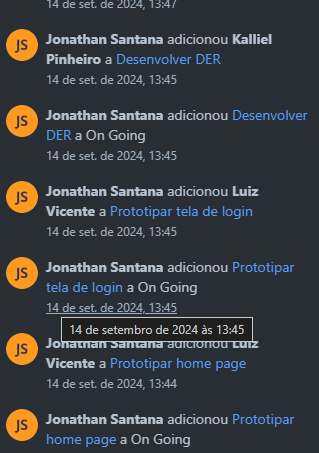
Front-end, e bem pouco o back-end. Já havia tido experiência com PostgreSQL. Mas, no entanto, pude estar entendendo o funcionamento do django, um framework nunca visto antes. Já com o front, tenho conseguido ampliar minha visão, e a estrutura do sistema a respeito quando se trata da interação humana com o sistema, pensando na melhor usabilidade que o usuário possa ter em nosso projeto. Também, fazendo otimizações, com base em feedbacks e testes feitos pelos docentes e por nós da equipe. Vivendo e aprendendo linguagens e estruturas de códigos, sistemas…

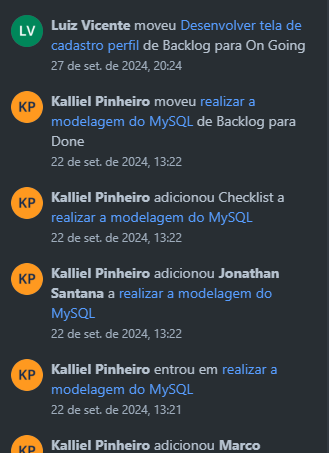
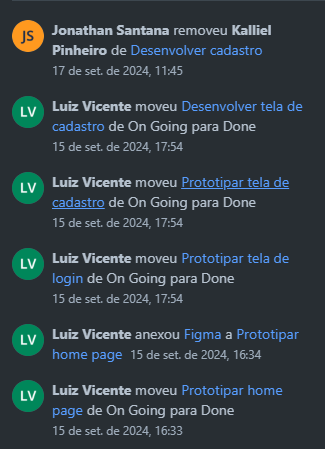
* Referências Bibliográficas:
* Documentação MongoDB
* Documentação Django
* Documentação Python
* Metodologia Scrum
* Anexo I - Diário de bordo

<https://trello.com/b/BRIcSQlk/pi-3-semestre>

* Anexo II – Cronograma efetivo







* Anexo III – Evidências



