Planejamento feito em 25/09 para o projeto 4 – Sistema de agendamento de consultas – Por José Alfredo Costa – UFRN

Aja como um profissional desenvolvedor experiente em Python e Django, além de scrum e gestão ágil em software. Queremos fazer a estrutura de sprints para um sistema que iremos desenvolver, com 4 participantes, durante os meses de outubro e novembro de 2024. Ajude-me a gerar a estrutura e descrições gerais das sprints semanais para o grupo e para cada pessoa, para o sistema:

“Projeto Integrador FAP: Sistema de Agendamento e Gerenciamento de Consultas para JA Serviços Médicos LTDA (SAGe) Resumo: O SAGe é um sistema web desenvolvido em Python e Django para agendamento de consultas e gerenciamento de pacientes para a clínica médica JA Serviços Médicos LTDA. A plataforma proporcionará aos pacientes uma maneira fácil e intuitiva de agendar consultas online, enquanto oferece à clínica uma ferramenta eficiente para organizar horários, gerenciar informações de pacientes e otimizar o fluxo de atendimento. Palavras-chave: Agendamento Online, Consultas Médicas, Python, Django, Gerenciamento de Clínicas, Prontuário Eletrônico.

1. Motivação Em um cenário médico cada vez mais digital, oferecer aos pacientes a conveniência do agendamento online e, ao mesmo tempo, otimizar o gerenciamento interno da clínica é crucial para o sucesso. Sistemas de agendamento tradicionais podem ser engessados e pouco eficientes, impactando a experiência do paciente e a produtividade da clínica. O SAGe surge para solucionar estes desafios, oferecendo uma plataforma moderna e integrada que automatiza o processo de agendamento, organiza os dados dos pacientes e facilita o acompanhamento das consultas.

2. Introdução O SAGe será desenvolvido utilizando Python e o framework Django, proporcionando uma solução robusta, escalável e segura para a JA Serviços Médicos LTDA. A plataforma será acessível via web, permitindo o agendamento de consultas por parte dos pacientes e o gerenciamento completo das informações por parte da clínica. O sistema contará com funcionalidades de CRUD para o cadastro de pacientes, médicos, especialidades e horários de atendimento. A interface intuitiva e amigável facilitará a navegação tanto para pacientes quanto para os funcionários da clínica.

3. Objetivo Geral Desenvolver um sistema web funcional, o SAGe, utilizando Python e o framework Django, para agendamento de consultas, gerenciamento de pacientes e controle de horários para a clínica médica JA

Serviços Médicos LTDA, otimizando o fluxo de atendimento e aprimorando a experiência do paciente.

4. Objetivos Específicos Agendamento de Consultas: Permitir que pacientes agendem consultas online, selecionando médico, especialidade, data e horário. Implementar um sistema de confirmação de consultas via e-mail e/ou SMS. Disponibilizar um portal do paciente para visualizar histórico de consultas, remarcar ou cancelar agendamentos. Gerenciamento de Pacientes: Implementar funcionalidades de CRUD para cadastro e gerenciamento de dados de pacientes (nome, contato, histórico médico, etc.). Permitir o upload e armazenamento de documentos médicos digitalizados. Implementar um sistema de busca e filtro para encontrar facilmente os dados dos pacientes. Gestão da Clínica: Permitir o cadastro e gerenciamento de médicos, especialidades e horários de atendimento. Gerar relatórios sobre consultas realizadas, pacientes atendidos, etc. Controlar o financeiro da clínica, com registro de pagamentos e emissão de relatórios. Implementar um sistema de controle de estoque de materiais (opcional). Funcionalidades Adicionais: Integrar o sistema com plataformas de videoconferência para teleconsultas (opcional). Implementar um sistema de lembretes de consultas para pacientes (opcional). Desenvolver um aplicativo mobile para pacientes acessarem o sistema (opcional).

5. Resultados Esperados Otimização do Tempo: Redução do tempo gasto com agendamentos por telefone, liberando a equipe para outras tarefas. Aumento da Eficiência: Melhor organização dos horários da clínica, redução de consultas não comparecidas e controle eficiente dos dados dos pacientes. Melhoria na Experiência do Paciente: Facilidade no agendamento de consultas, acesso online a informações e histórico médico, e melhor comunicação com a clínica. Redução de Custos: Diminuição de erros e retrabalhos, otimização de processos e melhor controle do estoque.

6. Equipe O desenvolvimento do SAGe será realizado por grupos de 4 a 5 alunos do curso FAP, sob a orientação de professores da UFRN e com a colaboração da equipe da JA Serviços Médicos LTDA. Descrição da Equipe: Desenvolvedor Back-End: Responsável pela lógica do sistema, banco de dados, segurança e implementação das funcionalidades. Desenvolvedor Front-End: Responsável pela interface do usuário, usabilidade, design responsivo e interação com o usuário. Analista de Banco de Dados: Responsável por modelar, criar e gerenciar o banco de dados do sistema. Analista de Testes: Responsável por planejar e executar testes para garantir a qualidade do sistema.

7. Bibliografia Básica

- Python Documentation. Disponível em: https://docs.python.org/

- Django Documentation. Disponível em: https://www.djangoproject.com/
- HTML, CSS & JavaScript. Disponível em: https://www.w3schools.com/
- Bootstrap Documentation. Disponível em: https://getbootstrap.com/
- PostgreSQL Documentation. Disponível em: https://www.postgresql.org/docs/

8. Plano de Desenvolvimento (Setembro - Dezembro 2024)

Sprint 1 (Setembro):
Levantamento de requisitos e definição do escopo do projeto. Modelagem do banco de dados e criação do modelo de dados. Configuração do ambiente de desenvolvimento e estrutura básica do projeto Django.

Sprint 2 (Outubro): Desenvolvimento das funcionalidades de CRUD para cadastro de pacientes, médicos e especialidades. Implementação da lógica de agendamento de consultas, incluindo verificação de disponibilidade de horários. Criação da interface para login e cadastro de usuários.

Sprint 3 (Novembro): Desenvolvimento da interface para agendamento de consultas por parte dos pacientes. Implementação do sistema de confirmação de consultas e lembretes. Integração com e-mail e/ou SMS.

Sprint 4 (Dezembro): Desenvolvimento do portal do paciente para visualizar histórico, remarcar e cancelar consultas. Implementação de funcionalidades adicionais (relatórios, controle financeiro, etc.). Testes de integração e aceitação do sistema.

9. Considerações Finais O SAGe tem potencial para transformar a gestão da JA Serviços Médicos LTDA, trazendo mais eficiência, organização e satisfação para pacientes e funcionários. A participação dos alunos do curso FAP nesse projeto oferecerá a oportunidade de aplicar seus conhecimentos em um sistema real, com impacto direto no dia a dia da clínica, contribuindo para o desenvolvimento de suas habilidades e para a criação de uma solução inovadora na área da saúde.”

Estrutura de Sprints para o Projeto SAGe

Baseado nesta estrutura de sprints, podemos ver que o desenvolvimento do SAGe está organizado em 8 sprints semanais, cobrindo o período de outubro a novembro de 2024, com uma semana adicional para finalização e entrega.

Cada sprint tem objetivos específicos e tarefas atribuídas a cada membro da equipe, garantindo que todas as áreas do projeto (back-end, front-end, banco de dados e testes) sejam desenvolvidas de forma paralela e integrada.

Estrutura de Sprints para o Projeto SAGe

Equipe

1. Desenvolvedor Back-End (DB)
2. Desenvolvedor Front-End (DF)
3. Analista de Banco de Dados (ABD)
4. Analista de Testes (AT)

Sprint 1 (30/09 - 06/10/2024): Planejamento e Configuração Inicial

Objetivos da Sprint:

- Definir requisitos detalhados
- Configurar ambiente de desenvolvimento
- Criar estrutura básica do projeto Django
- Iniciar modelagem do banco de dados

Tarefas por membro:

- DB: Configurar ambiente Django, criar estrutura básica do projeto
- DF: Pesquisar e propor design system, criar protótipos de baixa fidelidade
- ABD: Iniciar modelagem do banco de dados, definir entidades principais
- AT: Elaborar plano de testes, definir critérios de aceitação

Sprint 2 (07/10 - 13/10/2024): Desenvolvimento do Core

Objetivos da Sprint:

- Implementar modelos de dados básicos
- Criar views e templates iniciais
- Configurar autenticação de usuários

Tarefas por membro:

- DB: Implementar modelos Django para pacientes, médicos e especialidades
- DF: Criar templates base e implementar sistema de autenticação (front-end)
- ABD: Finalizar modelagem do banco de dados, criar migrations iniciais

- AT: Desenvolver testes unitários para modelos e autenticação

Sprint 3 (14/10 - 20/10/2024): Funcionalidades de CRUD

Objetivos da Sprint:

- Implementar CRUD completo para pacientes, médicos e especialidades
- Criar interfaces de usuário para operações CRUD

Tarefas por membro:

- DB: Desenvolver views e forms para operações CRUD
- DF: Criar interfaces de usuário para CRUD, implementar validações client-side
- ABD: Otimizar queries, criar índices necessários
- AT: Desenvolver testes de integração para operações CRUD

Sprint 4 (21/10 - 27/10/2024): Sistema de Agendamento

Objetivos da Sprint:

- Implementar lógica de agendamento de consultas
- Criar interface de agendamento para pacientes
- Desenvolver sistema de verificação de disponibilidade

Tarefas por membro:

- DB: Implementar lógica de agendamento e verificação de disponibilidade
- DF: Criar interface de agendamento, implementar calendário interativo
- ABD: Otimizar queries relacionadas ao agendamento
- AT: Desenvolver testes para o sistema de agendamento

Sprint 5 (28/10 - 03/11/2024): Notificações e Comunicação

Objetivos da Sprint:

- Implementar sistema de confirmação de consultas
- Desenvolver funcionalidade de lembretes
- Integrar com serviços de e-mail e SMS

Tarefas por membro:

- DB: Implementar lógica de notificações e integração com serviços externos
- DF: Criar interfaces para gerenciamento de notificações
- ABD: Modelar e implementar armazenamento de notificações
- AT: Testar sistema de notificações e integrações

Sprint 6 (04/11 - 10/11/2024): Portal do Paciente

Objetivos da Sprint:

- Desenvolver portal do paciente
- Implementar visualização de histórico de consultas
- Criar funcionalidades de remarcação e cancelamento

Tarefas por membro:

- DB: Implementar lógica do portal do paciente e histórico de consultas
- DF: Criar interface do portal do paciente, implementar visualizações de dados
- ABD: Otimizar queries para recuperação eficiente de histórico
- AT: Desenvolver testes de usabilidade e aceitação para o portal

Sprint 7 (11/11 - 17/11/2024): Relatórios e Dashboards

Objetivos da Sprint:

- Implementar geração de relatórios
- Criar dashboards para visualização de dados
- Desenvolver controle financeiro básico

Tarefas por membro:

- DB: Implementar lógica de geração de relatórios e cálculos financeiros
- DF: Criar interfaces para relatórios e dashboards
- ABD: Criar views e procedures para relatórios complexos
- AT: Testar precisão dos relatórios e cálculos financeiros

Sprint 8 (18/11 - 24/11/2024): Refinamento e Testes

Objetivos da Sprint:

- Realizar testes abrangentes do sistema
- Corrigir bugs e refinar funcionalidades
- Implementar melhorias de desempenho
- Preparar documentação final

Tarefas por membro:

- DB: Corrigir bugs, otimizar código, documentar API
- DF: Refinar interfaces, corrigir problemas de usabilidade, documentar front-end
- ABD: Otimizar consultas, criar índices adicionais, documentar estrutura do banco
- AT: Conduzir testes de sistema, documentar casos de teste e resultados

Entrega Final (25/11 - 30/11/2024):

- Finalizar documentação
- Preparar apresentação do projeto
- Realizar deploy do sistema em ambiente de produção

Alguns pontos importantes a considerar:

1. Flexibilidade: Esta estrutura serve como um guia, mas deve ser flexível para acomodar mudanças e desafios que possam surgir durante o desenvolvimento.
2. Reuniões diárias: Recomenda-se que a equipe realize reuniões diárias curtas (15 minutos) para alinhar o progresso e identificar obstáculos.
3. Revisão e retrospectiva: Ao final de cada sprint, é importante realizar uma reunião de revisão para demonstrar o que foi concluído e uma retrospectiva para discutir o que funcionou bem e o que pode ser melhorado.
4. Backlog de produto: Mantenha um backlog de produto atualizado com todas as funcionalidades desejadas, priorizando-as conforme necessário.
5. Integração contínua: Implemente práticas de integração contínua para detectar e resolver problemas de integração precocemente.
6. Comunicação com o cliente: Mantenha comunicação regular com a JA Serviços Médicos LTDA para garantir que o desenvolvimento esteja alinhado com suas expectativas e necessidades.

Descreva agora com detalhes o sprint 1 para o grupo e para cada pessoa, com códigos e diagramas, figuras, tabelas etc. seja o máximo tutorial

Sprint 1 Detalhado - Projeto SAGe

Sprint 1: Planejamento e Configuração Inicial (30/09 - 06/10/2024)

Objetivos da Sprint

1. Definir requisitos detalhados

2. Configurar ambiente de desenvolvimento
3. Criar estrutura básica do projeto Django
4. Iniciar modelagem do banco de dados

Kickoff do Sprint (30/09/2024)

Agenda da Reunião

1. Apresentação dos objetivos do sprint
2. Revisão do Product Backlog
3. Definição das tarefas para cada membro
4. Estabelecimento de critérios de aceitação
5. Planejamento das daily scrums

Tarefas Detalhadas por Membro

Desenvolvedor Back-End (DB)

1. Configurar ambiente Django

- Instalar Python 3.9+
- Criar ambiente virtual:

```
python -m venv sage_env
source sage_env/bin/activate  # No Windows:
sage_env\Scripts\activate
```

- Instalar Django e dependências:

```
pip install django psycopg2-binary
pip freeze > requirements.txt
```

2. Criar estrutura básica do projeto Django

- Iniciar projeto Django:

```
django-admin startproject sage_project
cd sage_project
python manage.py startapp core
python manage.py startapp appointments
python manage.py startapp users
```

- Configurar `settings.py`:

```
INSTALLED_APPS = [
    # ...
    'core',
    'appointments',
```

```
    'users',
]

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'sage_db',
        'USER': 'sage_user',
        'PASSWORD': 'sage_password',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Desenvolvedor Front-End (DF)

1. Pesquisar e propor design system

- Analisar opções como Material-UI, Bootstrap, ou Tailwind
- Criar documento comparativo com prós e contras de cada opção

2. Criar protótipos de baixa fidelidade

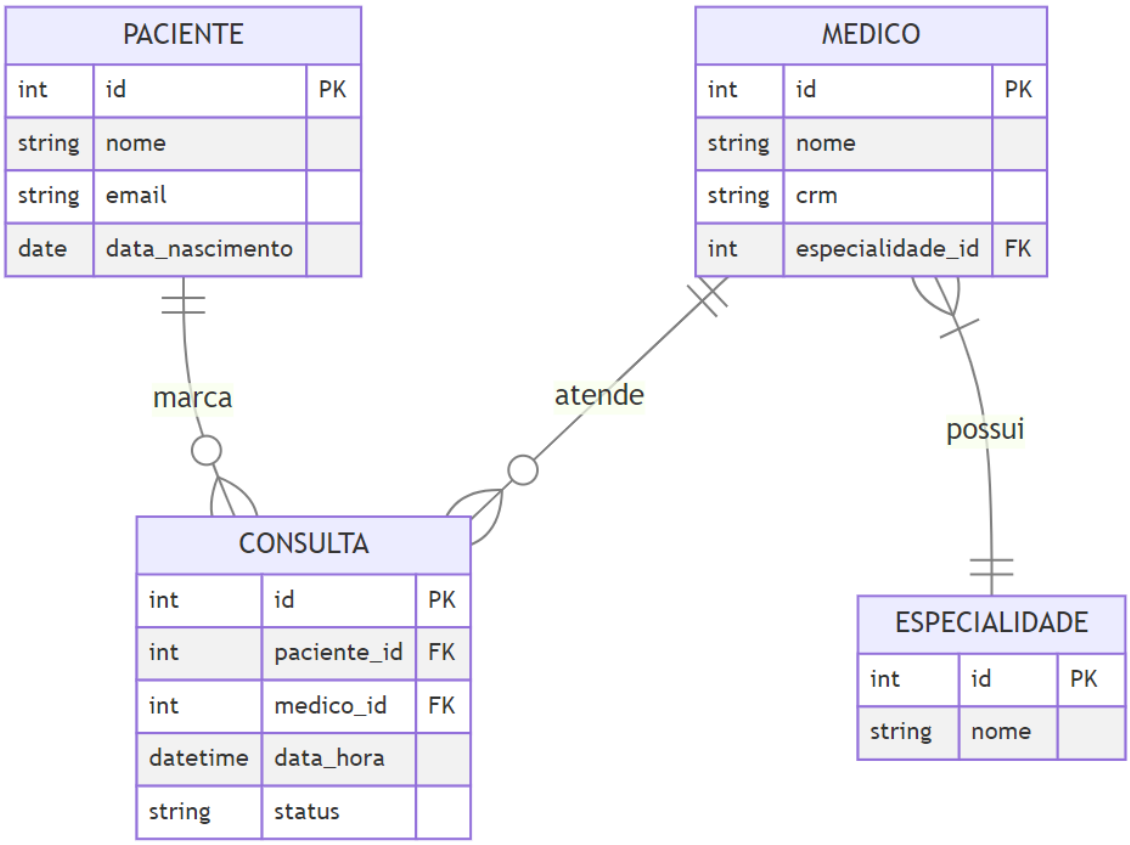
- Utilizar ferramentas como Figma ou Sketch
- Criar wireframes para:
    - Página inicial
    - Tela de login
    - Dashboard do paciente
    - Formulário de agendamento

Analista de Banco de Dados (ABD)

1. Iniciar modelagem do banco de dados

- Criar diagrama ER inicial usando ferramentas como Draw.io ou MySQL Workbench
- Definir entidades principais:
    - Paciente
    - Médico
    - Especialidade
    - Consulta
- Exemplo de diagrama ER simplificado:[1]

[1] https://mermaid.live/view#pako:eNqdU11rwjAU_SvhPlextdW2b6IdiJ-gexkFuTRRAzYpaQpz2v--1E5ddQy2PITec0_uOT0kJ0gkZRACUyOOO4VpLIhZy8FwHM3XETmfWy15IsPFfPU6XQ9ISFJUCdasWTQaDxc_cFAzQVmDVBrS-Uyi1TIajgfT8Wgwigwzk3le8Fg8yJ7gulpcaMIpWU7uUK4VFzsiZMqeQJYiP9xRaqxUG24E5glPmdCy7pYNe_8XTFTaPMvyjCUcD5wiZRsz6WXSUGxG8EfhrxXm3rH8_XkEZJtz8dcPJtZcyyhP52Kky0yaqOre9VPhkKNeoi_xmCSxImTLJU3OVLpZi0HtmTENoPinbYnHQMcSiNFQstFwdRQKhVgWzQMlit4dwi4fcVEVW6X_dxislQ_Em5fcSwhO8Q9hyum3f8btuz-33un4Q-J4FR4PbXjtwvH4n8LygY_cd2y0t-LjMsNsd23N7XcfvObZtu65vgQlCSzWrH8PlTZSfyqnnlA




```
erDiagram
    PACIENTE ||--o{ CONSULTA : marca
    MEDICO ||--o{ CONSULTA : atende
    MEDICO }|--|| ESPECIALIDADE : possui

    PACIENTE {
        int id PK
        string nome
        string email
        date data_nascimento
    }
    MEDICO {
        int id PK
        string nome
        string crm
        int especialidade_id FK
    }
    ESPECIALIDADE {
        int id PK
        string nome
    }
    CONSULTA {
        int id PK
        int paciente_id FK
        int medico_id FK
        datetime data_hora
        string status
```

```
    }
```

Analista de Testes (AT)

1. Elaborar plano de testes

- Criar documento com estratégia de testes, incluindo:
  - Testes unitários
  - Testes de integração
  - Testes de aceitação
  - Testes de desempenho

2. Definir critérios de aceitação

- Elaborar lista de critérios de aceitação para as principais funcionalidades
- Exemplo para a funcionalidade de agendamento:

ID	Critério de Aceitação
CA1	O paciente deve poder selecionar uma data disponível para consulta
CA2	O sistema deve mostrar apenas horários disponíveis para o médico selecionado
CA3	Após o agendamento, o paciente deve receber uma confirmação por e-mail
CA4	O médico deve ser notificado sobre novos agendamentos

Daily Scrums

- Horário: 09:00 - 09:15, todos os dias
- Local: Sala de reuniões virtual (link da videoconferência)
- Formato:
  1. O que foi feito ontem?
  2. O que será feito hoje?
  3. Há algum impedimento?

Revisão do Sprint (06/10/2024)

Agenda da Reunião

1. Demonstração das configurações e protótipos
2. Revisão do diagrama ER e plano de testes
3. Discussão sobre desafios enfrentados
4. Planejamento inicial para o próximo sprint

Retrospectiva do Sprint (06/10/2024)

Formato: Start, Stop, Continue

- Start: Que práticas deveríamos começar a fazer?

- Stop: Que práticas deveríamos parar de fazer?
- Continue: Que práticas estão funcionando bem e deveríamos continuar?

Próximos Passos

- Refinar o Product Backlog com base nos insights do Sprint 1
- Preparar-se para o desenvolvimento das funcionalidades core no Sprint 2

Este detalhamento do Sprint 1 fornece uma estrutura clara e orientações específicas para cada membro da equipe. Vamos analisar alguns pontos importantes:

1. Estrutura do Projeto: O Desenvolvedor Back-End (DB) está configurando uma estrutura Django sólida com apps separados para funcionalidades principais. Isso promove uma organização clara do código.
2. Design System: O Desenvolvedor Front-End (DF) está pesquisando opções de design system. Isso é crucial para manter consistência visual e acelerar o desenvolvimento da interface.
3. Modelagem de Dados: O Analista de Banco de Dados (ABD) iniciou com um diagrama ER simples, mas fundamental. Isso servirá como base para discussões e refinamentos futuros.
4. Plano de Testes: O Analista de Testes (AT) está estabelecendo uma estratégia de testes abrangente desde o início, o que é uma prática excelente para garantir a qualidade do software.
5. Daily Scrums: As reuniões diárias curtas ajudarão a manter todos alinhados e identificar problemas rapidamente.
6. Revisão e Retrospectiva: Estas sessões ao final do sprint são cruciais para avaliar o progresso e melhorar continuamente o processo de desenvolvimento.

Recomendações para a equipe:

1. Colaboração: Embora cada membro tenha tarefas específicas, é importante manter uma comunicação aberta. Por exemplo, o DF deve discutir os protótipos com o DB para garantir viabilidade técnica.
2. Documentação: Mantenha toda a documentação (diagramas, protótipos, planos de teste) em um local centralizado e acessível a todos.
3. Preparação para o Próximo Sprint: Use os últimos dias do Sprint 1 para refinar o backlog e preparar-se para começar o desenvolvimento no Sprint 2.

4. Aprendizado Contínuo: Incentive a equipe a compartilhar conhecimentos. Por exemplo, o DB pode fazer uma breve apresentação sobre a estrutura Django para todos.

Como poderíamos organizar as informações do grupo usando sistemas como Jira, de gestão de projetos ? Explique com detalhes

Sistema de gestão de projetos como o Jira é uma excelente maneira de manter o projeto estruturado, facilitar a colaboração e acompanhar o progresso. Vou explicar detalhadamente como poderíamos configurar o Jira para o projeto SAGe.

Configuração do Jira para o Projeto SAGe

Configuração do Jira para o Projeto SAGe

1. Criação do Projeto

1. Log in no Jira e clique em "Criar projeto".
2. Escolha "Software" como tipo de projeto.
3. Nomeie o projeto como "SAGe - Sistema de Agendamento e Gerenciamento".
4. Selecione "Scrum" como metodologia.

2. Configuração do Backlog

Epics

Crie Epics para representar as principais funcionalidades do sistema:

1. Autenticação e Autorização
2. Gerenciamento de Pacientes
3. Gerenciamento de Médicos
4. Agendamento de Consultas
5. Notificações e Lembretes
6. Relatórios e Dashboards

User Stories

Dentro de cada Epic, crie User Stories. Exemplo para o Epic "Agendamento de Consultas":

- Como paciente, quero poder ver os horários disponíveis para agendar uma consulta.
- Como paciente, quero poder selecionar um médico específico para minha consulta.
- Como médico, quero ver minha agenda de consultas para o dia/semana.

Tasks

Quebre cada User Story em tarefas específicas. Exemplo para "Ver horários disponíveis":

1. Implementar API para buscar horários disponíveis
2. Criar componente de calendário no front-end
3. Integrar componente de calendário com a API
4. Implementar filtros de busca (por data, médico, especialidade)

3. Configuração do Board

1. Crie um board Scrum para o projeto.
2. Configure as colunas do board:
    - To Do
    - In Progress
    - In Review
    - Done
3. Configure os swimlanes por membro da equipe:
    - Desenvolvedor Back-End
    - Desenvolvedor Front-End
    - Analista de Banco de Dados
    - Analista de Testes

4. Sprints

1. Crie sprints no Jira, começando com "SAGe Sprint 1".
2. Defina a duração do sprint (1 semana no nosso caso).
3. Adicione as User Stories e Tasks planejadas para o Sprint 1 ao sprint atual.

5. Estimativas e Pontos de História

1. Use a funcionalidade de estimativa do Jira para atribuir pontos de história às User Stories.
2. Recomenda-se usar a escala de Fibonacci (1, 2, 3, 5, 8, 13, 21).

6. Customização de Campos

Adicione campos customizados para melhorar o rastreamento:

1. Prioridade: Alta, Média, Baixa
2. Tipo de Tarefa: Feature, Bug, Melhoria

3. Componente: Back-end, Front-end, Banco de Dados, Testes

7. Automações

Configure algumas automações úteis:

1. Mover item para "In Review" quando um pull request for aberto.
2. Notificar o Analista de Testes quando um item for movido para "In Review".
3. Mover item para "Done" quando o pull request for mesclado.

8. Relatórios e Dashboards

Configure relatórios importantes:

1. Burndown Chart: Para visualizar o progresso do sprint.
2. Velocity Chart: Para entender a velocidade da equipe ao longo do tempo.
3. Cumulative Flow Diagram: Para identificar gargalos no processo.

Crie um dashboard personalizado com:

- Burndown do sprint atual
- Tarefas atribuídas a cada membro
- Bugs abertos
- Próximas User Stories no backlog

9. Integrações

Integre o Jira com outras ferramentas:

1. GitHub/GitLab: Para vincular commits e pull requests às tarefas.
2. Slack: Para notificações de atualizações importantes.
3. Confluence: Para vincular documentação detalhada às tarefas.

10. Permissões e Acessos

Configure as permissões adequadas:

1. Administrador do Projeto: Acesso total
2. Membros da Equipe: Criar e editar issues, mover cards no board
3. Stakeholders: Visualizar board e relatórios

11. Rituais Scrum no Jira

1. Sprint Planning: Use a funcionalidade de planejamento de sprint do Jira.
2. Daily Standup: Atualize o status das tarefas diariamente.
3. Sprint Review: Use o relatório de sprint para discutir o que foi concluído.
4. Sprint Retrospective: Crie uma página no Confluence vinculada ao sprint para documentar a retrospectiva.

12. Boas Práticas

1. Mantenha as descrições das tarefas claras e concisas.
2. Use labels para categorizar issues (ex: "frontend", "API", "UX").
3. Vincule issues relacionadas usando a funcionalidade de links do Jira.
4. Atualize regularmente o status das tarefas e o tempo gasto.
5. Use a funcionalidade de comentários para discussões relevantes à tarefa.

Essa configuração do Jira para o projeto SAGe fornece uma estrutura robusta para gerenciar o desenvolvimento do sistema. Vamos analisar alguns pontos-chave:

1. Estrutura Hierárquica: A organização em Epics > User Stories > Tasks permite uma visão clara do projeto em diferentes níveis de granularidade.
2. Visualização do Trabalho: O board Scrum com swimlanes por membro da equipe oferece uma visão clara de quem está trabalhando em quê.
3. Planejamento e Acompanhamento: A criação de sprints e o uso de estimativas ajudam no planejamento e no acompanhamento do progresso.
4. Customização: Campos personalizados e automações adaptam o Jira às necessidades específicas do projeto SAGe.
5. Métricas e Relatórios: Os relatórios configurados (Burndown, Velocity, etc.) fornecem insights valiosos sobre o desempenho da equipe e o andamento do projeto.
6. Integração: A conexão com outras ferramentas como GitHub e Slack melhora o fluxo de trabalho e a comunicação.

Recomendações para implementação:

1. Treinamento: Garanta que todos os membros da equipe estejam familiarizados com o Jira e com a configuração específica do projeto.
2. Consistência: Estabeleça diretrizes claras para a criação e atualização de issues para manter a consistência.
3. Revisão Regular: Faça revisões periódicas da configuração do Jira e ajuste conforme necessário com base no feedback da equipe.
4. Uso de Filtros: Crie filtros personalizados para ajudar cada membro da equipe a focar em suas tarefas relevantes.
5. Documentação: Mantenha uma página no Confluence ou documento similar com instruções sobre como usar o Jira para este projeto específico.
6. Iteração: Comece com uma configuração básica e refine ao longo do tempo com base nas necessidades da equipe.

Esta configuração do Jira proporcionará uma base sólida para o gerenciamento ágil do projeto SAGe. Ela facilitará a colaboração, o acompanhamento do progresso e a identificação precoce de problemas, contribuindo significativamente para o sucesso do projeto.

Para o sistema, vamos definir os requisitos funcionais e não funcionais, e toda documentação e planejamento necessários para a organização e início do projeto. Ajude-me com sua inteligência

Documentação e Planejamento do Projeto SAGe

Documentação e Planejamento do Projeto SAGe

1. Visão Geral do Projeto

O Sistema de Agendamento e Gerenciamento de Consultas (SAGe) é uma plataforma web desenvolvida para a JA Serviços Médicos LTDA, com o objetivo de otimizar o processo de agendamento de consultas e gerenciamento de pacientes.

1.1 Objetivos Principais

- Facilitar o agendamento de consultas para pacientes
- Melhorar a gestão de horários e agenda dos médicos
- Automatizar processos administrativos da clínica
- Proporcionar uma melhor experiência para pacientes e funcionários

2. Requisitos Funcionais

2.1 Gestão de Usuários

- RF01: O sistema deve permitir o cadastro de pacientes com informações pessoais e médicas básicas
- RF02: O sistema deve permitir o cadastro de médicos com suas respectivas especialidades e horários de atendimento
- RF03: O sistema deve suportar autenticação e autorização de usuários com diferentes níveis de acesso

2.2 Agendamento de Consultas

- RF04: Pacientes devem poder visualizar horários disponíveis e agendar consultas online

- RF05: O sistema deve permitir que pacientes cancelem ou remarquem consultas com antecedência mínima de 24 horas
- RF06: Médicos devem poder visualizar sua agenda de consultas

2.3 Gestão de Consultas

- RF07: O sistema deve enviar lembretes automáticos de consultas para pacientes via e-mail e/ou SMS
- RF08: Médicos devem poder registrar observações e diagnósticos após cada consulta
- RF09: O sistema deve gerar um histórico de consultas para cada paciente

2.4 Relatórios e Análises

- RF10: O sistema deve gerar relatórios de produtividade dos médicos
- RF11: O sistema deve fornecer análises sobre taxa de ocupação e cancelamentos

3. Requisitos Não Funcionais

3.1 Desempenho

- RNF01: O sistema deve suportar até 1000 usuários simultâneos sem degradação de performance
- RNF02: O tempo de resposta para operações de agendamento não deve exceder 3 segundos

3.2 Segurança

- RNF03: Todas as comunicações devem ser criptografadas usando HTTPS
- RNF04: O sistema deve estar em conformidade com a LGPD (Lei Geral de Proteção de Dados)

3.3 Usabilidade

- RNF05: A interface do usuário deve ser responsiva, funcionando em desktops, tablets e smartphones
- RNF06: O sistema deve ser acessível, seguindo as diretrizes WCAG 2.1 nível AA

3.4 Confiabilidade

- RNF07: O sistema deve ter um uptime de 99,9%
- RNF08: Backups completos devem ser realizados diariamente

3.5 Escalabilidade

- RNF09: O sistema deve ser capaz de escalar horizontalmente para acomodar crescimento futuro

4. Arquitetura do Sistema

4.1 Visão Geral da Arquitetura

O SAGe será desenvolvido como uma aplicação web utilizando uma arquitetura de três camadas:

1. Camada de Apresentação: Interface do usuário web
2. Camada de Aplicação: Lógica de negócios implementada em Django
3. Camada de Dados: Banco de dados PostgreSQL

4.2 Diagrama de Arquitetura

mermaid
Copy

```
graph TD
    A[Cliente Web] -->|HTTPS| B[Load Balancer]
    B --> C[Servidor Web]
    C --> D[Aplicação Django]
    D --> E[Banco de Dados PostgreSQL]
    D --> F[Servidor de E-mail]
    D --> G[Servidor de SMS]
    H[Serviço de Backup] --> E
```

5. Modelo de Dados

5.1 Diagrama Entidade-Relacionamento

mermaid
Copy

```
erDiagram
    PACIENTE ||--o{ CONSULTA : marca
    MEDICO ||--o{ CONSULTA : atende
    MEDICO }|--|| ESPECIALIDADE : possui
    PACIENTE {
        int id PK
        string nome
        string email
        date data_nascimento
        string telefone
    }
    MEDICO {
        int id PK
        string nome
        string crm
        int especialidade_id FK
    }
    ESPECIALIDADE {
        int id PK
        string nome
    }
```

```
CONSULTA {
    int id PK
    int paciente_id FK
    int medico_id FK
    datetime data_hora
    string status
    text observacoes
}
```

6. Planejamento do Projeto

6.1 Cronograma

Fase	Duração	Data de Início	Data de Término
Planejamento e Análise	2 semanas	30/09/2024	13/10/2024
Desenvolvimento	6 semanas	14/10/2024	24/11/2024
Testes e Correções	2 semanas	25/11/2024	08/12/2024
Implantação	1 semana	09/12/2024	15/12/2024

6.2 Equipe

- 1 Gerente de Projeto
- 2 Desenvolvedores Back-end
- 2 Desenvolvedores Front-end
- 1 Designer UX/UI
- 1 Analista de Qualidade
- 1 Administrador de Banco de Dados

6.3 Metodologia

O projeto será desenvolvido utilizando a metodologia Scrum, com sprints de uma semana.

7. Plano de Testes

7.1 Tipos de Testes

- Testes Unitários
- Testes de Integração
- Testes de Sistema
- Testes de Aceitação do Usuário (UAT)
- Testes de Desempenho
- Testes de Segurança

7.2 Critérios de Aceitação

- 100% dos testes unitários passando
- Cobertura de código de pelo menos 80%

- Todos os requisitos funcionais testados e aprovados
- Desempenho conforme especificado nos requisitos não funcionais

8. Plano de Implantação

8.1 Ambiente de Produção

- Servidor Web: Nginx
- Servidor de Aplicação: Gunicorn
- Banco de Dados: PostgreSQL
- Cache: Redis

8.2 Estratégia de Implantação

1. Preparação do ambiente de produção
2. Migração de dados (se aplicável)
3. Implantação da aplicação
4. Testes finais no ambiente de produção
5. Treinamento dos usuários
6. Go-live

9. Riscos e Mitigações

Risco	Probabilidade	Impacto	Mitigação
Atraso no desenvolvimento	Média	Alto	Priorização de funcionalidades, possível extensão do prazo
Problemas de integração	Média	Médio	Testes de integração contínuos, reuniões regulares de alinhamento técnico
Resistência dos usuários	Baixa	Alto	Envolvimento dos stakeholders desde o início, treinamento adequado
Problemas de segurança	Baixa	Alto	Revisões de código, testes de penetração, atualizações regulares de segurança

10. Considerações Finais

Este documento serve como base para o desenvolvimento do Sistema de Agendamento e Gerenciamento de Consultas (SAGe). Ele deve ser revisado e atualizado regularmente conforme o projeto avança e novas informações se tornam disponíveis.

Esta documentação abrangente fornece uma base sólida para o início do projeto SAGe. Vamos analisar alguns pontos-chave:

1. Requisitos Claros: Os requisitos funcionais e não funcionais estão bem definidos, fornecendo uma direção clara para o desenvolvimento.

2. Arquitetura: A visão geral da arquitetura e o diagrama ajudam a entender como o sistema será estruturado.
3. Modelo de Dados: O diagrama ER oferece uma visão clara das entidades principais e seus relacionamentos.
4. Planejamento: O cronograma, a composição da equipe e a metodologia estão bem definidos.
5. Testes e Implantação: Os planos de teste e implantação fornecem um roteiro para garantir a qualidade e o lançamento bem-sucedido do sistema.
6. Riscos: A identificação precoce de riscos e estratégias de mitigação ajuda a preparar a equipe para possíveis desafios.

Recomendações para o próximo passo:

1. Revisão com Stakeholders: Apresente esta documentação aos principais stakeholders da JA Serviços Médicos LTDA para garantir que está alinhada com suas expectativas.
2. Priorização de Requisitos: Trabalhe com os stakeholders para priorizar os requisitos e definir o MVP (Produto Mínimo Viável).
3. Configuração do Ambiente: Comece a configurar o ambiente de desenvolvimento com base na arquitetura definida.
4. Definição de Sprints: Use esta documentação para planejar os primeiros sprints, focando inicialmente nos requisitos de maior prioridade.
5. Estabelecimento de Padrões: Defina padrões de codificação, nomenclatura e documentação para o projeto.
6. Treinamento da Equipe: Garanta que todos os membros da equipe estejam familiarizados com as tecnologias e metodologias escolhidas.
7. Criação de Protótipos: Comece a criar protótipos de interface para as funcionalidades principais.

Esta documentação serve como um excelente ponto de partida para o projeto SAGe. À medida que o desenvolvimento avança, certifique-se de manter este documento atualizado com quaisquer mudanças ou decisões importantes.

Faça agora o sprint 2 - para o grupo e para cada indivíduo, como faz para o 1, e também todos os diagramas, tabelas e possibilidades e códigos e explicações possíveis

Sprint 2 Detalhado - Projeto SAGe

Sprint 2: Desenvolvimento do Core (07/10 - 13/10/2024)

Objetivos do Sprint

1. Implementar modelos de dados básicos
2. Criar views e templates iniciais
3. Configurar autenticação de usuários
4. Iniciar desenvolvimento das funcionalidades CRUD básicas

Kickoff do Sprint (07/10/2024)

Agenda da Reunião

1. Revisão dos objetivos do sprint
2. Análise do backlog e priorização de tarefas
3. Distribuição das tarefas entre os membros da equipe
4. Discussão sobre possíveis desafios e soluções

Tarefas Detalhadas por Membro

Desenvolvedor Back-End (DB)

1. Implementar modelos Django

```python
# core/models.py
from django.db import models
from django.contrib.auth.models import User

class Especialidade(models.Model):
    nome = models.CharField(max_length=100)

    def __str__(self):
        return self.nome

class Medico(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    crm = models.CharField(max_length=20, unique=True)
    especialidade = models.ForeignKey(Especialidade,
on_delete=models.SET_NULL, null=True)

    def __str__(self):
        return f"Dr. {self.user.get_full_name()} ({self.crm})"

class Paciente(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    data_nascimento = models.DateField()
```

```
    telefone = models.CharField(max_length=20)

    def __str__(self):
        return f"{self.user.get_full_name()} ({self.user.email})"

class Consulta(models.Model):
    paciente = models.ForeignKey(Paciente, on_delete=models.CASCADE)
    medico = models.ForeignKey(Medico, on_delete=models.CASCADE)
    data_hora = models.DateTimeField()
    status = models.CharField(max_length=20, choices=[
        ('AGENDADA', 'Agendada'),
        ('CONFIRMADA', 'Confirmada'),
        ('REALIZADA', 'Realizada'),
        ('CANCELADA', 'Cancelada')
    ])
    observacoes = models.TextField(blank=True, null=True)

    def __str__(self):
        return f"Consulta: {self.paciente} com {self.medico} em
{self.data_hora}"
```

2. Criar views básicas

```
# core/views.py
from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from .models import Paciente, Medico, Consulta

@login_required
def lista_consultas(request):
    if hasattr(request.user, 'paciente'):
        consultas =
Consulta.objects.filter(paciente=request.user.paciente)
    elif hasattr(request.user, 'medico'):
        consultas =
Consulta.objects.filter(medico=request.user.medico)
    else:
        consultas = Consulta.objects.all()
    return render(request, 'core/lista_consultas.html', {'consultas':
consultas})

@login_required
def detalhe_consulta(request, consulta_id):
    consulta = get_object_or_404(Consulta, id=consulta_id)
    return render(request, 'core/detalhe_consulta.html', {'consulta':
consulta})

# Adicione mais views conforme necessário
```

3. Configurar URLs

```python
# core/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('consultas/', views.lista_consultas, name='lista_consultas'),
    path('consulta/<int:consulta_id>/', views.detalhe_consulta,
name='detalhe_consulta'),
    # Adicione mais URLs conforme necessário
]
```

4. Configurar autenticação

```python
# settings.py (adicione estas linhas)
LOGIN_REDIRECT_URL = '/consultas/'
LOGOUT_REDIRECT_URL = '/'

# project/urls.py (adicione estas linhas)
from django.contrib.auth import views as auth_views

urlpatterns = [
    # ...
    path('login/', auth_views.LoginView.as_view(), name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
    # ...
]
```

Desenvolvedor Front-End (DF)

1. Criar templates base

```python
<!-- templates/base.html -->
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>{% block title %}SAGe{% endblock %}</title>
    <link
href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.mi
n.css" rel="stylesheet">
</head>
<body class="bg-gray-100">
    <nav class="bg-blue-600 p-4 text-white">
        <div class="container mx-auto flex justify-between">
            <a href="/" class="font-bold text-xl">SAGe</a>
            <div>
                {% if user.is_authenticated %}
                    <a href="{% url 'logout' %}" class="ml-4">Sair</a>
                {% else %}
```

```
                    <a href="{% url 'login' %}" class="ml-
4">Entrar</a>
                {% endif %}
            </div>
        </div>
    </nav>
    <main class="container mx-auto mt-8 p-4">
        {% block content %}
        {% endblock %}
    </main>
</body>
</html>
```

2. Criar template para lista de consultas

```
<!-- templates/core/lista_consultas.html -->
{% extends 'base.html' %}

{% block title %}Minhas Consultas{% endblock %}

{% block content %}
<h1 class="text-2xl font-bold mb-4">Minhas Consultas</h1>
<div class="grid gap-4">
    {% for consulta in consultas %}
    <div class="bg-white p-4 rounded shadow">
        <h2 class="font-bold">{{ consulta.medico }}</h2>
        <p>Data: {{ consulta.data_hora|date:"d/m/Y H:i" }}</p>
        <p>Status: {{ consulta.get_status_display }}</p>
        <a href="{% url 'detalhe_consulta' consulta.id %}"
class="text-blue-600 hover:underline">Ver detalhes</a>
    </div>
    {% empty %}
    <p>Você não tem consultas agendadas.</p>
    {% endfor %}
</div>
{% endblock %}
```

3. Implementar formulário de login

```
<!-- templates/registration/login.html -->
{% extends 'base.html' %}

{% block title %}Login{% endblock %}

{% block content %}
<div class="max-w-md mx-auto bg-white p-8 rounded shadow">
    <h2 class="text-2xl font-bold mb-4">Login</h2>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
```

```
        <button type="submit" class="bg-blue-600 text-white px-4 py-2
rounded mt-4">Entrar</button>
    </form>
</div>
{% endblock %}
```

Analista de Banco de Dados (ABD)

1. Refinamento do modelo de dados

- Revisar e ajustar o modelo ER conforme necessário
- Criar índices para otimização de consultas frequentes

sql

```
-- Exemplo de criação de índices
CREATE INDEX idx_consulta_data_hora ON consulta(data_hora);
CREATE INDEX idx_medico_especialidade ON medico(especialidade_id);
```

2. Implementar migrations

```
python manage.py makemigrations
python manage.py migrate
```

3. Criar dados de teste

```
# core/management/commands/criar_dados_teste.py
from django.core.management.base import BaseCommand
from django.contrib.auth.models import User
from core.models import Especialidade, Medico, Paciente, Consulta
from datetime import datetime, timedelta

class Command(BaseCommand):
    help = 'Cria dados de teste para o sistema SAGe'

    def handle(self, *args, **options):
        # Criar especialidades
        esp_cardiologia =
Especialidade.objects.create(nome='Cardiologia')
        esp_pediatria = Especialidade.objects.create(nome='Pediatria')

        # Criar médicos
        user_medico1 = User.objects.create_user('drsilva',
'drsilva@example.com', 'senha123')
        medico1 = Medico.objects.create(user=user_medico1,
crm='12345SP', especialidade=esp_cardiologia)

        user_medico2 = User.objects.create_user('drasantos',
'drasantos@example.com', 'senha123')
        medico2 = Medico.objects.create(user=user_medico2,
crm='67890SP', especialidade=esp_pediatria)
```

```
        # Criar pacientes
        user_paciente1 = User.objects.create_user('joao',
'joao@example.com', 'senha123')
        paciente1 = Paciente.objects.create(user=user_paciente1,
data_nascimento='1980-01-01', telefone='11999999999')

        user_paciente2 = User.objects.create_user('maria',
'maria@example.com', 'senha123')
        paciente2 = Paciente.objects.create(user=user_paciente2,
data_nascimento='1990-05-15', telefone='11988888888')

        # Criar consultas
        data_hora = datetime.now() + timedelta(days=1)
        Consulta.objects.create(paciente=paciente1, medico=medico1,
data_hora=data_hora, status='AGENDADA')

        data_hora = datetime.now() + timedelta(days=2)
        Consulta.objects.create(paciente=paciente2, medico=medico2,
data_hora=data_hora, status='AGENDADA')

        self.stdout.write(self.style.SUCCESS('Dados de teste criados
com sucesso!'))
```

Analista de Testes (AT)

1. Desenvolver testes unitários

```
# core/tests.py
from django.test import TestCase
from django.contrib.auth.models import User
from .models import Especialidade, Medico, Paciente, Consulta

class ModelTestCase(TestCase):
    def setUp(self):
        self.especialidade =
Especialidade.objects.create(nome='Cardiologia')
        self.user_medico = User.objects.create_user('drsilva',
'drsilva@example.com', 'senha123')
        self.medico = Medico.objects.create(user=self.user_medico,
crm='12345SP', especialidade=self.especialidade)
        self.user_paciente = User.objects.create_user('joao',
'joao@example.com', 'senha123')
        self.paciente =
Paciente.objects.create(user=self.user_paciente,
data_nascimento='1980-01-01', telefone='11999999999')

    def test_especialidade_str(self):
        self.assertEqual(str(self.especialidade), 'Cardiologia')

    def test_medico_str(self):
        self.assertEqual(str(self.medico), 'Dr. drsilva (12345SP)')
```

```
    def test_paciente_str(self):
        self.assertEqual(str(self.paciente), 'joao
(joao@example.com)')

    def test_consulta_creation(self):
        consulta = Consulta.objects.create(
            paciente=self.paciente,
            medico=self.medico,
            data_hora='2024-10-10 10:00:00',
            status='AGENDADA'
        )
        self.assertIsNotNone(consulta.id)
        self.assertEqual(consulta.status, 'AGENDADA')

# Adicione mais testes conforme necessário
```

2. Configurar testes de integração

```
# core/tests_integration.py
from django.test import TestCase
from django.urls import reverse
from django.contrib.auth.models import User
from .models import Paciente, Medico, Consulta

class ViewTestCase(TestCase):
    def setUp(self):
        self.user = User.objects.create_user('testuser',
'test@example.com', 'testpassword')
        self.paciente = Paciente.objects.create(user=self.user,
data_nascimento='1990-01-01', telefone='11999999999')

    def test_lista_consultas_view(self):
        self.client.login(username='testuser',
password='testpassword')
        response = self.client.get(reverse('lista_consultas'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'core/lista_consultas.html')

    # Adicione mais testes de integração conforme necessário
```

Daily Scrums

- Horário: 09:00 - 09:15, todos os dias
- Local: Sala de reuniões virtual (link da videoconferência)
- Formato:
    1. O que foi feito ontem?
    2. O que será feito hoje?
    3. Há algum impedimento?

Revisão do Sprint (13/10/2024)

Agenda da Reunião

1. Demonstração das funcionalidades implementadas
2. Análise do cumprimento dos objetivos do sprint
3. Coleta de feedback da equipe e stakeholders
4. Identificação de pontos de melhoria

Retrospectiva do Sprint (13/10/2024)

Formato: Start, Stop, Continue

- Start: Que práticas deveríamos começar a fazer?
- Stop: Que práticas deveríamos parar de fazer?
- Continue: Que práticas estão funcionando bem e deveríamos continuar?

Métricas do Sprint

Métrica	Valor
Pontos concluídos	X
Velocidade da equipe	Y pontos/sprint
Bugs encontrados	Z
Cobertura de testes	W%

Diagrama de Burndown do Sprint

```
mermaid
graph TD
    A[Sprint Start] -->|Day 1| B(20 points)
    B -->|Day
```