

Projeto Le Menu

Para a comunidade acadêmica da UNIFESSPA que precisa de acesso prático às informações do restaurante, o Le Menu é uma plataforma web que facilita consulta de cardápio, comunicação e feedback, diferente de processos manuais, nosso sistema centraliza todas as informações em um local acessível 24/7.

Sequenciador



Figura 1: Sequenciador

Product backlog

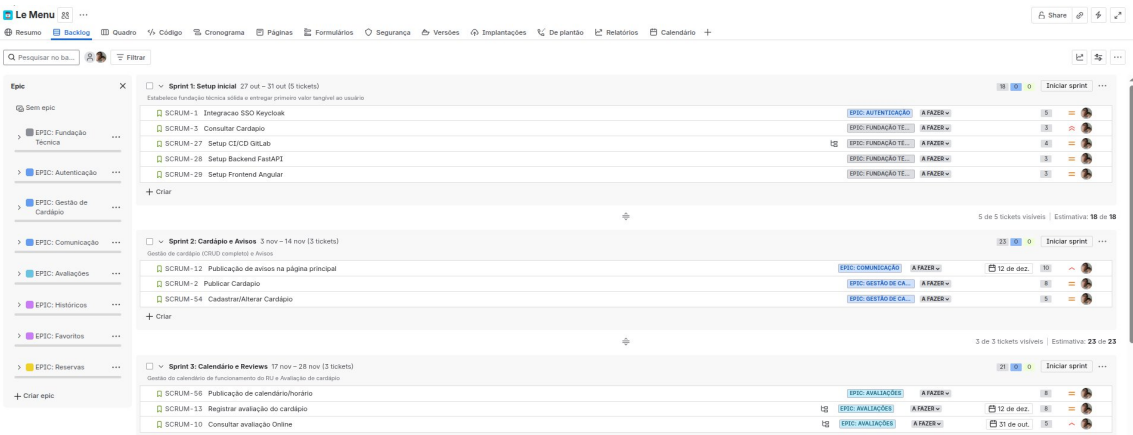


Figura 2: Backlog do MVP



Figura 3: Backlog dos incrementos

Detalhamento do Sprint 1

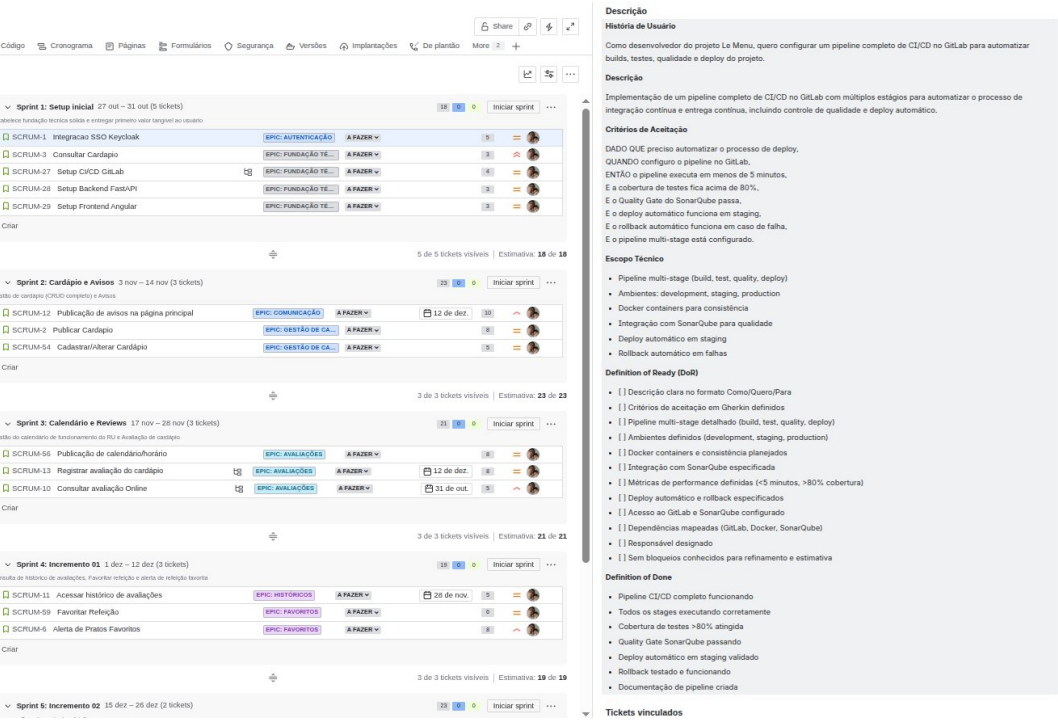


Figura 4: Sprint 1: Integração SSO Keycloak

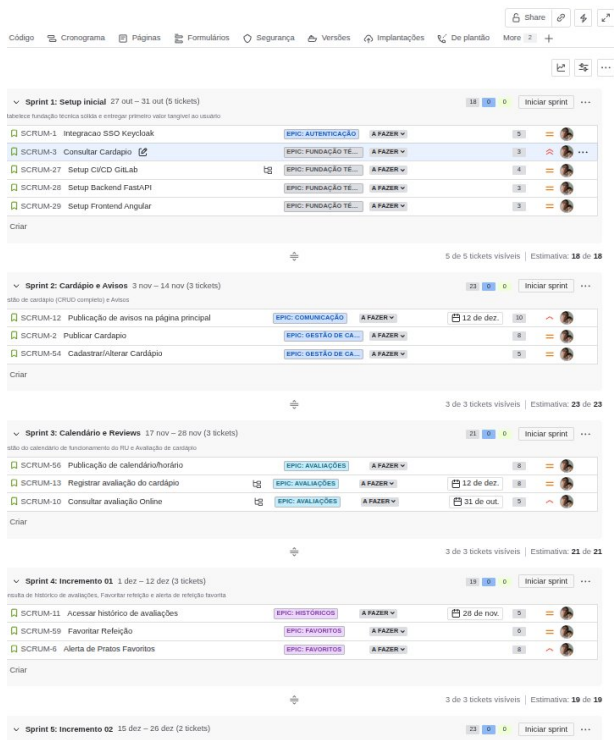


Figura 5: Sprint 1: Consultar Cardápio

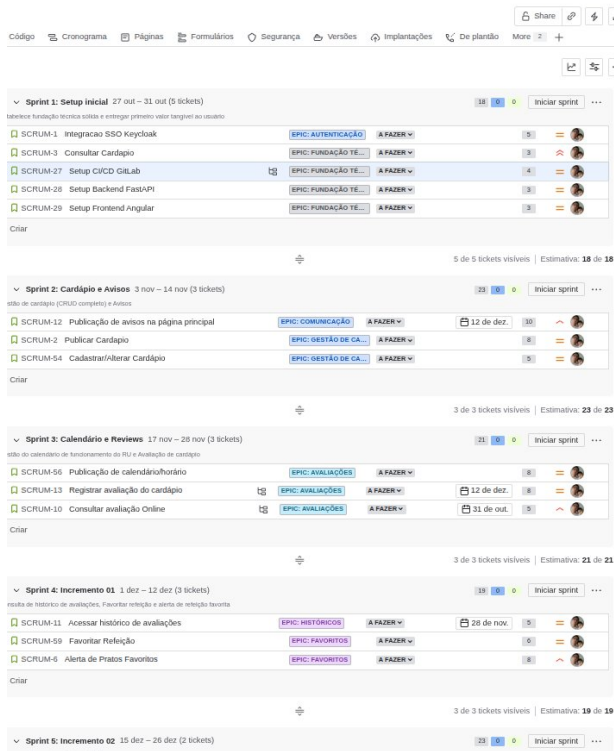


Figura 6: Sprint 1: Setup CI/CD GitLab

SCRUM-10 / SCRUM-3

Consultar Cardápio

A fazer +

Descrição

História de usuário

Como membro da comunidade acadêmica

Quero visualizar o cardápio da semana de forma publica

Para planejar minhas refeições no RU sem precisar fazer login

Descrição

Versão inicial do sistema de consulta de cardápio, focando na entrega rápida de valor. Permite visualização pública do cardápio semanal sem autenticação.

Critérios de aceitação

- DADO QUE acesso à URL de homologação
- QUANDO a página carrega
- ENTÃO vejo o cardápio da semana atual
- E visualizo informações básicas das refeições
- E no mobile o layout se adapta responsivamente
- E se não há cardápio publicado, vejo "Cardápio em elaboração"

Definition of Ready (DoR)

- [] Descrição clara no formato Como/Quero/Para
- [] Critérios de aceitação em Gherkin definidos
- [] Interface de consulta publica especificada
- [] Design responsivo para mobile definido
- [] Requisitos de performance especificados (< 2s de carregamento, < 500ms API)
- [] Critérios de acessibilidade definidos (WCAG 2.1 AA >= 90%)
- [] Mensagem de "Cardápio em elaboração" especificada
- [] Cobertura de testes definida (>= 70%)
- [] URL de homologação disponível
- [] Dependências mapeadas (Postgres, Frontend, CI/CD)
- [] Responsável designado
- [] Sem bloqueios conhecidos para refinamento e estimativa

Definition of Done

- Interface de consulta implementada
- Design responsivo funcionando
- Performance: Tempo de carregamento < 2s
- API Response: Endpoints < 500ms
- Acessibilidade: Score WCAG 2.1 AA >= 90%
- Testes unitários passando (>= 70% de cobertura)
- Validação do Product Owner

Tickets vinculados

- SCRUM-10: Integracao SSO Keycloak
- SCRUM-3: Consultar Cardápio
- SCRUM-27: Setup CI/CD GitLab
- SCRUM-28: Setup Backend FastAPI
- SCRUM-29: Setup Frontend Angular

SCRUM-44 / SCRUM-47

Outras informações

Escopo Técnico

- Pipeline multi-stage (build, test, quality, deploy)
- Ambientes: development, staging, production
- Docker containers para consistência
- Integração com SonarQube para qualidade
- Deploy automático em staging
- Rollback automático em falhas

Definition of Ready (DoR)

- [] Descrição clara no formato Como/Quero/Para
- [] Critérios de aceitação em Gherkin definidos
- [] Pipeline multi-stage detalhado (build, test, quality, deploy)
- [] Ambientes definidos (development, staging, production)
- [] Docker containers e consistência planejados
- [] Integração com SonarQube especificada
- [] Métricas de performance definidas (>5 minutos, >80% cobertura)
- [] Deploy automático e rollback especificados
- [] Acesso ao GitLab e SonarQube configurado
- [] Dependências mapeadas (GitLab, Docker, SonarQube)
- [] Responsável designado
- [] Sem bloqueios conhecidos para refinamento e estimativa

Definition of Done

- Pipeline CI/CD completo funcionando
- Todos os stages executando corretamente
- Cobertura de testes >80% atingida
- Quality Gate SonarQube passando
- Deploy automático em staging validado
- Rollback testado e funcionando
- Documentação de pipeline criada

Tickets filhos

0/1 concluído

Ticket	Prioridade	Responsavel	Status
SCRUM-44 T1: Configurar arquivo .gitlab-ci.yml	Medium	Jefferson Ferreira	A FAZER
SCRUM-45 T2: Setup Docker build stage	Medium	Jefferson Ferreira	A FAZER
SCRUM-46 T3: Configurar test stage	Medium	Jefferson Ferreira	A FAZER
SCRUM-47 T4: Setup deploy automatico	Medium	Jefferson Ferreira	A FAZER

Tickets vinculados

Adicionar ticket vinculado

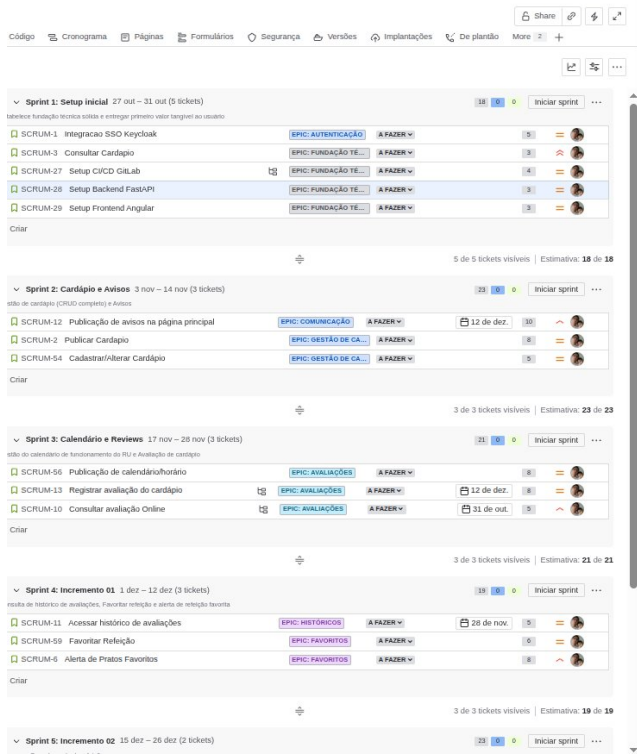


Figura 7: Sprint 1: Setup Backend FastAPI

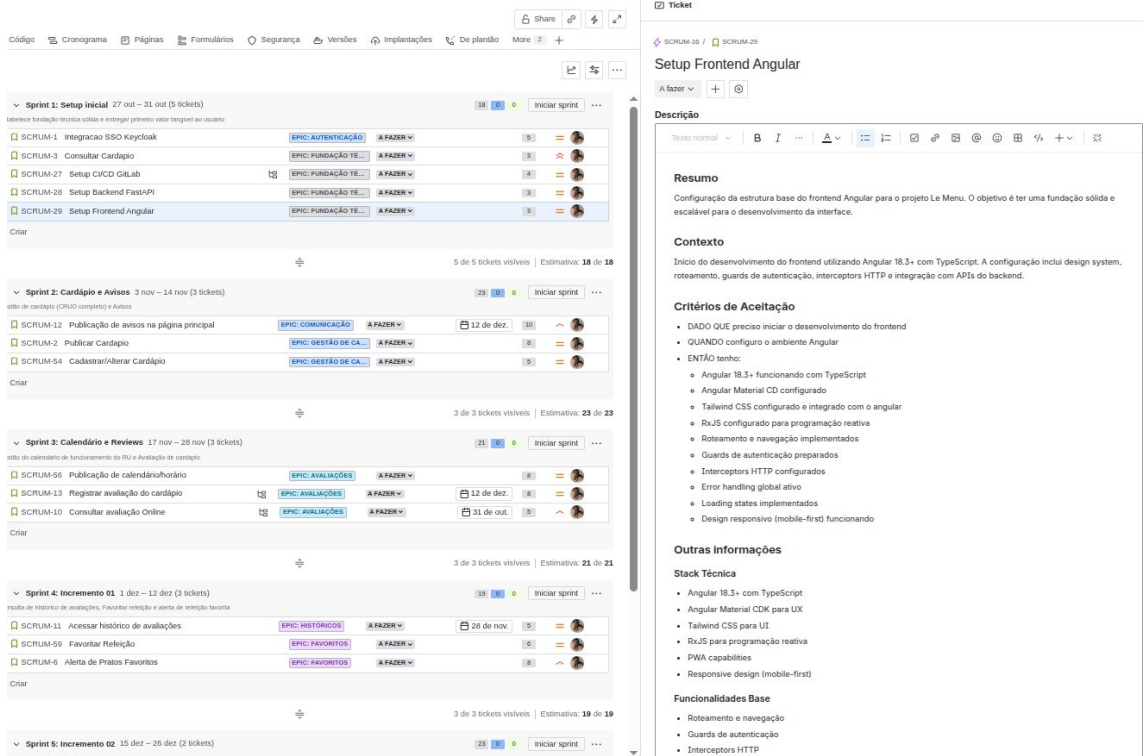


Figura 8: Sprint 1: Setup Frontend Angular

Critérios de Aceitação

- DADO QUE preciso iniciar o desenvolvimento do backend
- QUANDO configuro o ambiente de desenvolvimento
- ENTÃO tenho:
 - FastAPI funcionando com Python 3.13+
 - PostgreSQL conectado via SQLAlchemy
 - Alembic configurado para migrations
 - Poetry gerenciando dependências
 - Estrutura de pastas organizada
 - Autenticação JWT preparada para Keycloak

Outras Informações

Arquitetura Técnica

- FastAPI + Python 3.13+
- SQLAlchemy ORM + Alembic migrations
- Banco de dados PostgreSQL
- Pydantic para validação
- Poetry para dependências

Estrutura de Pastas

- app/core/ - Configurações e utilitários
- app/api/v1/ - Endpoints da API
- app/models/ - Modelos SQLAlchemy
- app/services/ - Lógica de negócio

Definition of Ready (DoR)

- [] Descrição clara no formato Como/Quero/Para
- [] Critérios de aceitação em Gherkin definidos
- [] Stack tecnológico especificado (FastAPI, Python 3.13+, PostgreSQL)
- [] Estrutura de pastas detalhada
- [] SQLAlchemy ORM e Alembic migrations planejados
- [] Pydantic para validação especificado
- [] Poetry para gerenciamento de dependências definido
- [] Autenticação JWT preparada para Keycloak
- [] Configurações e utilitários planejados
- [] Dependências mapeadas (PostgreSQL, Poetry, FastAPI)
- [] Responsável designado
- [] Sem bloqueios conhecidos para refinamento e estimativa

Definition of Done

- Estrutura base do backend implementada
- FastAPI funcionando com endpoints básicos
- PostgreSQL conectado com SQLAlchemy
- Alembic migrations configurado
- Poetry dependencies gerenciadas
- Pydantic validation funcionando
- JWT auth preparado para Keycloak
- Testes básicos implementados

Observação: Não adicionamos capturas de telas de todas os sprints devido o tamanho final do documento gerado. Todos os PDF exportados do projeto Le Menu estarão no mesmo repositório deste arquivo.

Cronograma

O cronograma do projeto foi estruturado em cinco Sprints. O primeira, com duração de uma semana, é dedicado ao setup inicial e às configurações de CI/CD. Ainda assim, já haverá entrega de valor, disponibilizando a página inicial do sistema com o cardápio da semana de implantação, carregado diretamente no banco de dados.

Os sprints seguintes foram planejados com duração de duas semanas cada. O MVP será entregue no terceiro Sprint, restando mais dois incrementos posteriores. Durante todo o período de implementação e incrementos, serão coletados feedbacks de usuários e stakeholders para orientar o planejamento da segunda fase do projeto, que abrangerá integrações com outros sistemas, além de melhorias de desempenho e usabilidade.

A terceira e última fase terá como foco a integração com o sistema de controle de entrada do restaurante, permitindo o gerenciamento automático de pontos e saldos por meio da leitura do QR Code nas catracas do restaurante.

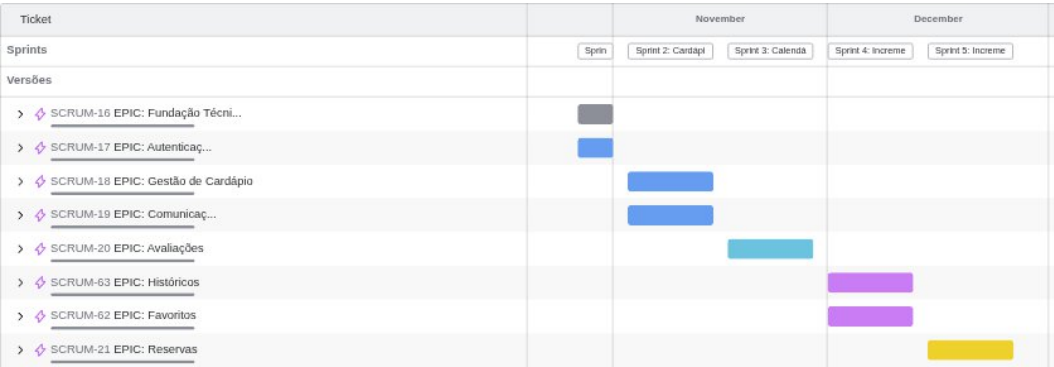


Figura 9: Cronograma do projeto com seis sprints