

CRONOGRAMA DE DESENVOLVIMENTO – JAVA ADVANCED

Projeto: Pedix – Comanda Digital Inteligente

**Integrantes: Alane Rocha da Silva, Anna Beatriz de Araujo Bonfim,
Maria Eduarda Araujo Penas**

Sprint 1 – Entregas Concluídas (Java)

Responsável: Alane Rocha

Itens Entregues

- Definição do domínio inicial (ItemCardápio, Pedido, PedidoItem)
- Estruturação da API com camadas Controller, Service e Repository
- Implementação de endpoints CRUD seguindo REST nível 1
- Configuração de persistência com Oracle, JPA e Hibernate
- Execução dos testes no Postman com cenários funcionais

Status geral da Sprint 1: Concluído

**

Sprint 2 – Entregas Concluídas (Java)

Responsável: Alane Rocha

Itens Entregues

- Implementação de HATEOAS (maturidade nível 3)
- Refatoração das entidades e serviços (DTOs, validações, encapsulamento)
- Ampliação dos testes no Postman (cenários 200/400/404)
- Atualização da documentação e diagramas no repositório
- Levantamento dos pontos de integração futura com a API .NET

Data da Sprint 2: 09/11/2025

Sprint 3 – Planejamento (Java)

O escopo oficial ainda será divulgado. Abaixo está um resumo objetivo do que já foi entregue e do que será desenvolvido quando o novo conteúdo for liberado.

Backlog Proposto – Sprint 3 (Java)

Domínio / Classes

- **CategoriaCardápio** (entidade + relacionamento com ItemCardápio)
- **Extensões de ItemCardápio** (entidade já existente; adicionar flags de destaque, disponibilidade, tempo de preparo)
- **DTOs de Relatório** (ex.: RelatorioConsumoDTO, RankingItensDTO)
- **ConfigRelatorios** (parâmetros persistidos para períodos, filtros e agrupamentos)

Integração com API .NET (read-only)

- Client HTTP interno para **consultar mesas e comandas** e compor relatórios
- Endpoints agregadores em Java para expor **relatórios baseados em dados da .NET** (sem duplicar CRUD)

Endpoints / Contratos

- GET /cardapio | GET /cardapio/categorias
- GET /relatorios/consumo?periodo=
- GET /relatorios/itens-mais-vendidos?periodo=
- Padrões de **paginação e filtros** (page, size, sort, categoria)

Suporte ao Mobile

- App passa a **consumir cardápio real** da API Java
- Cache leve e tratamento de ausência de rede no consumo do cardápio
- Padronização de respostas para simplificar formulários

Dados e Qualidade

- Índices e ajustes de consulta no Oracle para endpoints de leitura
- Coleção Postman atualizada em /docs/testes (cenários de sucesso e erro)
- OpenAPI/Swagger revisado e versionado

Responsável pela Sprint 3: Anna Beatriz

Sprint 4 – Planejamento (Java)

O escopo oficial ainda será divulgado. Abaixo está um resumo objetivo do que já foi entregue e do que será desenvolvido quando o novo conteúdo for liberado.

Backlog Proposto – Sprint 4 (Java)

Relatórios e Métricas Avançadas

- Rotatividade de mesas por período
- Ticket médio e horários de pico
- Dashboards administrativos

Integrações com .NET

- Ampliação das consultas read-only para relatórios consolidados
- Melhorias de resiliência no client HTTP

Performance e Manutenibilidade

- Paginação, filtros e ordenação padronizados
- Otimizações de consulta e projeções JPA
- Cache seletivo para cardápio e métricas

Suporte ao Mobile

- Ajustes nas respostas para telas de relatórios
- Formatos adequados para gráficos e tabelas

Entrega Final

- Documentação consolidada
- Fluxo Mobile → .NET → Java → Banco de Dados
- Evidências em /docs

Responsável pela Sprint 4: Maria Eduarda

Expansão Prevista de Classes e Funcionalidades (Java)

Lista de Entregas Prováveis nas Próximas Etapas nas Próximas Etapas

Estas atividades representam a continuidade natural do projeto e refletem o tipo de evolução esperada em uma API Java secundária que dará suporte ao app mobile e ao ecossistema .NET.

1. Ampliação do Domínio Java

- Inclusão de categorias para o cardápio (ex.: bebidas, pratos principais, sobremesas)
- Estruturação de modelos para relatórios e métricas

- Extensão de `ItemCardápio` com atributos adicionais (ex.: tempo de preparo, destaque, disponibilidade)

2. Relatórios e Métricas Operacionais

- Rotas para itens mais consumidos por período
- Relatórios de horários de pico
- Indicadores para ticket médio
- Métricas de rotatividade por mesa (dados vindos da .NET)

3. Integração com API .NET (Consulta e Composição de Dados)

- Conector interno para leitura de comandas e mesas
- Composição de informações para dashboards administrativos
- Endpoints agregados: cardápio + status do restaurante

4. Melhoria da Qualidade e Manutenibilidade

- Implementação de paginação, filtros e ordenação em todas as consultas
- Preparação de respostas padronizadas (DTOs)
- Documentação ampliada com diagramas

5. Suporte ao App Mobile

- Endpoints otimizados para cardápio (cache e retorno leve)
- Rotas específicas para relatórios utilizados no app
- Estabilidade e padronização das respostas para facilitar integração

6. Preparação do Pacote Final do Projeto

- Revisão geral do repositório
- Consolidação de documentação, evidências e testes
- Diagrama completo do fluxo Mobile → .NET → Java → Banco de Dados

Propósito desta Expansão

A inclusão desses pontos reforça o planejamento técnico e apresenta ao professor uma visão estruturada da evolução do backend Java, evidenciando:

- clareza sobre responsabilidades entre Java e .NET;
- maturidade no planejamento das próximas etapas;
- compreensão das necessidades do app mobile;
- preparo para escalabilidade do sistema.

O detalhamento específico de cada entrega será organizado assim que o escopo oficial das próximas sprints for publicado. (Java)

Contexto: esta expansão não faz parte das Sprints 1 e 2. Será organizada quando o novo escopo for divulgado. Abaixo está o direcionamento alinhado ao papel do **mobile** e à divisão **API primária (.NET)** x **API secundária (Java)**.

1. Novas Classes (Java como API de Suporte)

Foco em leitura, catálogo e analytics — evitando duplicar o fluxo operacional da .NET.

- **ItemCardapio** (extensões; entidade já existente): atributos opcionais para filtros e destaque
- **RelatorioConsumo**: projeção/DTO para responder consultas agregadas
- **MétricasOperacionais**: agregações para dashboards (picos por horário, ticket médio)
- **ConfigRelatorios**: parâmetros salvos de relatórios e períodos

Observação: entidades **Mesa**, **Comanda** e **Pedido** permanecem **operacionais na .NET**. A API Java pode **consultá-las** quando necessário para compor relatórios, sem replicar o CRUD.

2. Integrações Previstas (Java ↔ .NET)

- **Leitura de comandas/mesas** via client interno para compor relatórios (read-only)
- **Cardápio**: entrega estável para o mobile (com cache e paginação)
- **Relatórios**: consumo por período, itens mais vendidos, rotatividade de mesas
- **Dashboards administrativos**: endpoints para painéis do gestor

3. Diretrizes para o Mobile

- O app consulta **.NET** para ações em tempo real (abrir comanda, enviar pedido, status)
- O app consulta **Java** para **cardápio** e **relatórios**
- Evitar que o mobile dependa de duas APIs na mesma ação — simplifica formulários e navegação