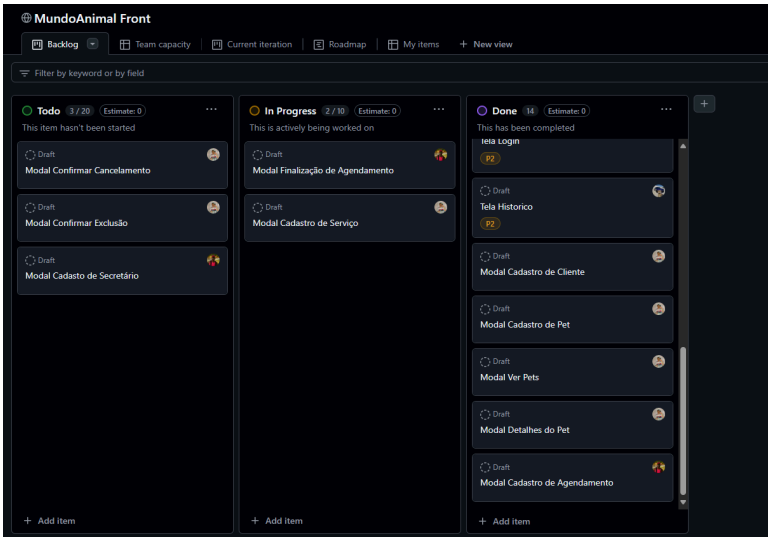
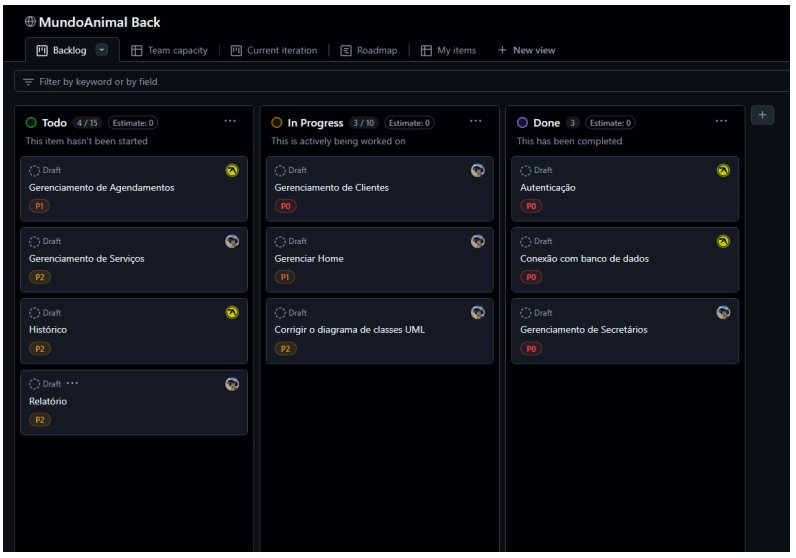
Diário de Bordo

O desenvolvimento do sistema Mundo Animal seguiu um processo iterativo e incremental baseado na metodologia Scrum. O time foi dividido em quatro áreas principais: UX, Requisitos, Front e Back, e fizemos questão de manter uma comunicação constante com a participação simultânea de membros em mais de uma área, garantindo flexibilidade e eficiência.

As *dailys* eram nosso momento chave para isso, realizadas nas segundas, quartas e sextas. Elas duravam poucos minutos, mas faziam toda a diferença para garantir que todos soubessem o que estava acontecendo e para resolver qualquer impedimento. Organizamos nosso trabalho em *sprints*, com metas bem definidas. Cada integrante sabia exatamente o que precisava entregar, mas o legal foi ver como nós ajudávamos quando alguém tinha dificuldade. Durante as sprints, o backlog foi refinado e priorizado, garantindo que as funcionalidades essenciais fossem entregues no tempo previsto.





Uma coisa que gostamos foi como o Scrum deixou o processo menos caótico. Sempre que tínhamos as *retrospectivas*, podíamos avaliar o que deu certo e o que precisava melhorar. Isso trouxe um sentimento de evolução contínua no projeto e na forma como trabalhávamos como equipe. No final, cada pedaço do sistema se conecta de forma bem natural, e isso foi graças ao alinhamento que o Scrum nos proporcionou.

Tomamos as seguintes decisões:

1. Optamos por utilizar exclusivamente **JavaFX**, abandonando a ideia inicial de combinar com Spring Boot. Essa decisão foi motivada por uma análise que indicou a viabilidade de simplificar a arquitetura sem comprometer as funcionalidades desejadas.
2. A equipe foi organizada para desenvolvermos o sistema de forma paralela, aproveitando assim o máximo de cada um dos membros.
3. O **Discord** foi escolhido como canal principal, por proporcionar uma melhor comunicação entre os integrantes da equipe e por poder dividir a comunicação em canais diferentes de acordo com as suas seguintes finalidades.
4. Decidimos usar o **PostgreSQL** devido à sua robustez e suporte para operações complexas necessárias no projeto.
5. A metodologia **Scrum** foi escolhida para facilitar a adaptação às mudanças frequentes e para garantir entregas incrementais.
6. Optamos por não usar o **Docker**. Após várias tentativas e erros de implementação, infelizmente não conseguimos.

Usamos as seguintes tecnologias:

1. Front-end/Interface: **Scenebuilder** para a construção das telas de acordo com o protótipo.
2. Back-end: Implementações integradas ao **JavaFX** na qual utiliza a linguagem de programação Java.
3. Banco de Dados: **PostgreSQL** para a persistência e gerenciamento dos dados do sistema.
4. **Git e GitHub** para o versionamento e hospedagem do nosso código para melhor compartilhamento entre os membros da equipe.
5. Comunicação entre os membros: **Discord.**
6. Prototipação do sistema: **Figma.**

Algumas ferramentas de apoio que utilizamos para o desenvolvimento além das tecnologias já mencionadas:

1. **IntelliJ IDEA:** IDE principal para o desenvolvimento da nossa aplicação.
2. **pgAdmin:** Gerenciador do banco de dados do Postgres.

Decisões tomadas pela equipe até agora:

1. União dos requisitos "Check-in de animais" e "Check-out de animais" em um único requisito: "Registro de entrada e saída de animais". Esse requisito ficou em Stand by por algumas semanas, pois nos reunimos com a nossa cliente só no dia 6 de dezembro para checarmos como realmente funciona.
2. Adicionamos um novo caso de uso "Cadastrar Serviço", na qual o Admin vai cadastrar o nome do serviço, uma descrição e o valor do serviço.
3. O Caso de Uso “Cadastrar Animal” se tornou “Cadastrar Cliente” depois de nos reunirmos com a nossa cliente e chegarmos a uma conclusão.
4. A partir da reunião com a professora de Banco de Dados (Lívia), levantamos algumas questões e fizemos mudanças como:
    a. Apagar a entidade Histórico e Relatório do nosso diagrama do banco de dados.
    b. Corrigir relacionamento de Agendamento com Serviço (1 para muitos).
5. Achamos melhor mudar a tecnologia Jira (gerenciamento de tarefas) para o Projects no GitHub, já que estamos utilizando-o para controle de versões, preferimos unificar em apenas uma plataforma nossas atividades.