UNIVERSIDADE FEDERAL DO CEARÁ

Relato de Experiência

Este documento visa compartilhar as experiências dos membros da equipe ao longo do desenvolvimento do Sistema de Gerência para Casa de Saberes Cego Aderaldo para a disciplina de Projeto Integrado em Engenharia de Software I, no semestre 2023-2, incluindo detalhes da execução do projeto e feedback das tecnologias usadas, do processo de desenvolvimento da aplicação e dos outros artefatos.

Equipe

- Francisco Paulino - 538451 - paulinofilho@alu.ufc.br
- Gustavo Henrique - 535735 - gustavohenriquefs.dev@gmail.com
- João Pedro - 539012 - joaopedroph@alu.ufc.br
- Robson Diógenes - 521437 - robsonad07@alu.ufc.br

Feedback

Esta seção apresenta o feedback de cada membro da equipe acerca dos principais artefatos utilizados ao longo do projeto, incluindo frameworks, APIs, SGBDs, ferramentas, entre outros artefatos.

Processos

- Trello
    - **Paulino -** Facilita muito a questão da organização do projeto, mas depende 100% de atualizações manuais feitas pelos membros da equipe, isso deixa a ferramenta um pouco isolada da parte do desenvolvimento **3,5/5**
    - **Gustavo -** Solução rápida e fácil para organizar pequenos projetos. 3/5
    - **João Pedro -** Criação de tarefas eficiente e a possibilidade de acompanhar a cronologia destas é interessante, mas a descrição das atividades acaba ficando muito afastada do código, integração com o github é possível, mas ferramentas como jira realizam melhor. **3/5**
    - **Robson -** Muito util para realizar a organização do projeto, mas falta suporte para gerenciamento de projetos mais complexos ou de grande escala. **3/5**

- Github Projects
    - **Paulino -** Solucionou de maneira satisfatória o principal problema do trello, pois permite linkar atividades registradas com atividades de desenvolvimento realizadas por algum membro da equipe. **5/5**
    - **Gustavo -** Uma solução mais integrada com o restante do projeto, facilitando o link entre a task e o restante dos dados do projeto. **5/5**
    - **João Pedro -** Mapeamento de criação de features, correções de erro e refatorações diretamente no repositório ajudam a manter todo o processo centralizado em um local. **5/5**
    - **Robson -** Facilitou muito mais o nosso trabalho, pois facilitou muito a gerência de todo o projeto, como citado pelo João Pedro, a ferramenta permitiu mapeamento de criação de features, correções de erro e refatorações diretamente no repositório 5/5

Tecnologias

- maven
    - **Paulino -** Permitiu de maneira mais fácil a integração das demais tecnologias como o JavaFX, sceneBuilder, dependências e variáveis de ambiente. **4,5/5**
    - **Gustavo -** Solução rápida e prática para o uso de dependências, porém não é tão prático. **4/5**

    - **João Pedro -** Adição de dependências simples e eficiente, dispensa completamente instalação local de jars, mas muitos comandos geram uma saída no terminal bastante poluída. **4/5**
    - **Robson -** Ferramenta muito útil, pois permitiu o gerenciamento de todas dependências do projeto, assim facilitando a integração de toda a equipe com o projeto. **4/5**
- JavaFX
    - **Paulino -** Framework bem completo que facilitou muito a parte do front-end do projeto, mas a estrutura do código era um tanto complexa. **4/5**
    - **Gustavo -** Apresenta muitos recursos para o desenvolvimento de interface. **4/5**
    - **João Pedro -** Relativamente simples de desenvolver, mas a sintaxe de bindings, listeners e algumas features que utilizam lambdas é bastante extensa, utilização de tabelas também é bastante complexa. **3/5**
    - **Robson -** É um framework de utilização relativamente simples que atende às expectativas estabelecidas. 3/5
- Junit
    - **Paulino -** Não tive contato com essa tecnologia.
    - **Gustavo -** Não tive contato.
    - **João Pedro -** Framework bastante completo, mas fácil de iniciar. O maior problema está nos testes para a camada DAO, sendo difícil garantir que o banco de testes volte ao estado original após a execução do testes. Por fim, logs de teste falhos mostram apenas erros de asserção, e não exceções levantadas durante a execução do teste. **3/5**
    - **Robson -** Não tive contato com essa tecnologia.
- PostgreSQL
    - **Paulino -** Além de dar suporte para arquivos multimídia, também permitiu a tradução de código SQL para o diagrama relacional, ganhando muito tempo para a equipe. **5/5**
    - **Gustavo -** Um SGBD bem robusto e com muitos recursos. **5/5**
    - **João Pedro -** A criação do esquema foi bastante simples, praticamente não necessitamos usar alguma sintaxe específica desse SGBD e a ferramenta ERDTool facilitou a criação do diagrama do esquema relacional. **5/5**
    - **Robson** - O SGBD é altamente eficaz, especialmente por oferecer suporte a arquivos multimídia, um aspecto fundamental para o funcionamento do nosso sistema. **5/5**
- Driver JDBC
    - **Paulino -** Não tive contato com essa tecnologia.
    - **Gustavo -** Apresenta muita documentação sobre ele, porém não transmite muita segunda. **2/5**
    - **João Pedro -** O driver é fácil de configurar, mas tem diversas desvantagens: executar a query diretamente é algo repetitivo e propenso a erros, atualizações no banco geram mudanças em diversas classes e não existe suporte à tipos personalizados. **2/5**
    - **Robson** - O driver é de fácil utilização, porém, sua característica de executar queries diretamente pode aumentar a propensão a erros. 3/5
- sdk java para AWS S3
    - **Paulino -** Não tive contato com essa tecnologia.
    - **Gustavo -** Documentação e código bem descritivo e completo. **5/5**

- **João Pedro -** Conexão com a instância bastante simples. Todas as principais operações que realizamos podem ser feitas facilmente usando os métodos já disponíveis. **5/5**
- **Robson -** Não tive contato com essa tecnologia.

Ferramentas

- Figma
  - **Paulino -** Ferramenta que foi utilizada durante todo o projeto e permitiu uma prototipação rápida e fácil para todos os protótipo, tanto de alto fidelidade quanto de baixa fidelidade. 5/5
  - **Gustavo -** Ferramenta com muitos recursos para o desenvolvimento de protótipos. **5/5**
  - **Jhordanna -**
  - **João Pedro** - O desenvolvimento dos protótipos de baixa e alta fidelidade foi bastante fluido, com um esquema de cores definido e um design system a seguir. Variantes e componentes ajudaram bastante na réplica de interações complexas, e variáveis são úteis em casos bastante específicos. Os plugins de compatibilidade com javaFX foram insuficientes para o nosso caso. **4,5/5**
  - **Robson** - A ferramenta é muito útil, uma vez que possibilita a criação de protótipos de alta fidelidade, tornando mais fácil a programação das telas do sistema.
- SceneBuilder
  - **Paulino -** Essa tecnologia permitiu grande optimização do processo de desenvolvimento das telas da aplicação. Além de permitir melhor compreensão da hierarquia do código FXML. **5/5**
  - **Gustavo -** Uso muito simples, mas não apresenta muitas opções de componentes. **4/5**
  - **João Pedro -** A ferramenta torna bastante fácil a criação das views necessárias, mas é difícil de manter um layout consistente e não existe uma forma simples de prever como a tela reagirá à interação do usuário, mesmo com uma classe controller definida. **4/5**
  - **Robson -** A ferramenta facilita a criação de views do sistema, porém é muito difícil criar um layout responsivo. **3.5/5**
- ElephantSQL
  - **Paulino -** Muito limitado por fornecer poucas conexões simultâneas com o banco de dados. 3/5
  - **Gustavo -** Serviço gratuito, mas com recursos muito limitados. **3/4**
  - **João Pedro -** Serviço fácil de acessar e configurar, mas disponibiliza pouco armazenamento e apenas 5 conexões simultâneas no plano gratuito. **3,5/5**
  - **Robson -** Serviço fácil de utilizar e configurar, e nos ajudou por deixar o sistema online, porém por estarmos utilizando a versão gratuita o seu uso foi bem limitado**. 4/5**
- Astah UML
  - **Paulino -** Bem completo para a construção de diagramas, mas pouco intuitivo na parte de diagrama de sequência. 4,5/5
  - **Gustavo -** Não tive contato.

    - **João Pedro -** Permite gerar diagramas de classe automaticamente a partir de código java importado, mas a criação de diagramas de sequência ainda é bastante complicada. **4/5**
    - **Robson -** Ferramenta muito útil para criação de diagramas e também muito fácil de se utilizar. **4/5**

O que melhorar

Alguns pontos que necessitam da maior atenção em uma possível continuidade do projeto são:

- Aplicar técnicas de ORM, possivelmente substituindo o driver JDBC por outras opções, como o framework Hibernate
- Tratamento de exceções sem silenciamento (não realizar apenas log)
- Refatoração do código para permitir que diferentes equipamentos possam gerenciar suas próprias informações pela mesma aplicação,
- Encapsular outros tipos de serviço de mídia, como Firebase.
- Conteinerização da aplicação
- Adição de funcionalidades de auditoria (histórico de alteração e permissões de usuários)
- Integração com API do Mapas Culturais para coletar e enviar informações de entidades e ações.