RELATÓRIO DE DESENVOLVIMENTO

PROJETO E-COMMERCE -AVALIAÇÃO AP003

TIME TIME-03

DATA 28/05/2025

EQUIPE E PAPÉIS

ALBERT SILVA DE JESUS - Líder Técnico
LAIO RODRIGUES - Desenvolvedor
LEANDRO GARCIA M. CERQUEIRA - Desenvolvedor
CARLOS ANDRÉ DE SOUZA DIAS - Desenvolvedor
DANIEL MONTEIRO MOTA - Desenvolvedor
BIANCARDY LIMA - Desenvolvedor
DIOGO DOREA - Desenvolvedor

JUSTIFICATIVA DE ALTERAÇÕES EM MÓDULOS

As alterações solicitadas pela empresa Techie envolveram a migração do foco de vendas de relógios e celulares para servidores da Dell e Positivo. Com isso, foi necessário realizar uma série de atualizações e reorganizações nos módulos do sistema para atender às novas demandas. A seguir, detalhamos as principais alterações realizadas e as justificativas técnicas para cada decisão tomada.

Reorganização de Pacotes e Classes

Pacote de Infraestrutura (infra):

- **Config:**

    - **CacheConfig:** Implementamos essa classe para melhorar o desempenho da aplicação ao reduzir o tempo de acesso a dados frequentemente solicitados. Isso é essencial para uma aplicação que lida com grandes volumes de dados, como a de vendas de servidores.
    - **SpringJpaAuditingConfig:** Adicionamos esta classe para facilitar a auditoria automática de entidades JPA, registrando quem criou ou modificou uma entidade e quando isso ocorreu. A integração com o sistema de segurança para obter o usuário autenticado garante a rastreabilidade e conformidade com políticas de segurança.

    - **SpringTimezoneConfig:** Configuramos a zona horária padrão da JVM para "America/Sao_Paulo", garantindo que todas as operações temporais na aplicação sejam realizadas nesta zona horária. Isso evita a necessidade de manipulação manual de fusos horários em várias partes do código, facilitando a manutenção e

prevenindo erros relacionados ao tempo.

Pacote Documentation:

- **SpringDocOpenApiConfig:** Configuramos o OpenAPI para a aplicação, definindo informações gerais da API e configurando o esquema de segurança para autenticação JWT. Isso melhora a documentação e a usabilidade da API, facilitando a integração com outros sistemas e o trabalho dos desenvolvedores.

Pacote Exception:

- **ApiExceptionHandler:** Centralizamos o tratamento de exceções específicas nesta classe, fornecendo respostas HTTP apropriadas e mensagens de erro consistentes aos clientes da API. Isso melhora a experiência do usuário ao lidar com erros.

- **ErrorMessage:** Criamos esta classe para permitir a criação de respostas de erro detalhadas e consistentes, melhorando a clareza das mensagens de erro e facilitando a depuração por parte dos desenvolvedores.

Pacote JWT:

- **Classes de Geração de Tokens:** Adicionamos várias classes para gerar tokens JWT, permitindo a troca segura de informações entre partes. Após a atualização do JWT, o sistema ficou mais seguro, eliminando uma vulnerabilidade que poderia causar vazamento de dados de clientes.

Pacote Security:

- **Atualização do Spring Security:** Após a atualização do Spring Security, o sistema ficou protegido contra ameaças e vulnerabilidades, garantindo a confidencialidade, integridade e disponibilidade dos dados e recursos. A integração do JWT com o Spring Security oferece uma robusta segurança para gerenciar autenticação e autorização na ShoppingStore.

Pacote DTO:

- **Pacotes View e Form:** Utilizamos records em vez de classes para os DTOs. Os forms

recebem dados das requisições dos usuários e as views retornam informações sem os dados sensíveis dos usuários. O uso de records simplifica a definição de classes que são usadas principalmente para transportar dados, reduzindo o código boilerplate e promovendo a imutabilidade.

Pacote Mapper:

- **Pacotes Forms, Updates, Views e Interface Mapper:** Definimos um contrato para converter objetos entre diferentes tipos (Input e Output). Esta estrutura promove a separação de preocupações e a reutilização de código, essencial para a manutenção e evolução da aplicação.

Pacote Enums:

- **PaymentStatusEnum e PaymentTypeEnum:** Implementamos esses enums para representar o estado e o tipo de pagamento na aplicação. Isso aumenta a clareza do código e ajuda a garantir que os valores utilizados sejam válidos, reduzindo a possibilidade de erros e permitindo uma manipulação segura e clara dos dados de pagamento.

Migrations com Flyway:

Implementamos o Flyway para facilitar a gestão de mudanças no esquema do banco de dados, garantindo a consistência e confiabilidade das alterações em ambientes de desenvolvimento, teste e produção. O versionamento das mudanças facilita o controle das alterações e a colaboração entre a equipe de desenvolvimento, assegurando que todos os ambientes estejam sincronizados e que as atualizações sejam aplicadas de forma ordenada.

Justificativa Técnica

As alterações foram motivadas pela necessidade de adaptar a aplicação às novas demandas de vendas de servidores, garantindo melhor desempenho, segurança e manutenibilidade. A reorganização dos pacotes e a introdução de novas configurações e tratamentos de exceções proporcionam uma base robusta e escalável para futuras expansões. A integração de tecnologias modernas, como JWT e Flyway, assegura que a aplicação esteja alinhada com as melhores práticas de desenvolvimento de software.

Benefícios das Alterações

1. **Desempenho:** Melhoria no acesso a dados frequentemente solicitados com a implementação de CacheConfig.
2. **Segurança:** Adoção de JWT e atualização do Spring Security aumentaram a proteção contra vulnerabilidades.
3. **Manutenção:** Simplificação da manipulação de fusos horários e tratamento de exceções centralizado, facilitando a manutenção do sistema.
4. **Documentação:** Melhoria na usabilidade da API com a configuração do OpenAPI.
5. **Escalabilidade:** Estrutura de mapper e DTO promove a separação de preocupações, facilitando a adição de novas funcionalidades.
6. **Consistência:** Uso de Flyway para gerenciamento de migrações de banco de dados, garantindo consistência e versionamento.

Conclusão

As alterações realizadas na aplicação da Techie foram essenciais para atender às novas demandas de negócios e melhorar a infraestrutura tecnológica. A reorganização dos pacotes, melhorias de desempenho e segurança, e a implementação de novas funcionalidades proporcionaram uma base sólida para o crescimento e a escalabilidade do sistema, alinhando a aplicação com as melhores práticas do mercado.

RELATÓRIO DE ATIVIDADES

TAREFA	RESPONSÁVEL	TEMPO PREVISTO	TEMPO EXECUTADO
Criação do repository GitHub	Albert	5 minutos	4 minutos
Spring initialize	Laio	5 minutos	4 minutos
Estrutura do projeto: organização (Pakage)	Laio	30 minutos	25 minutos
Modelagem do SGBD inicial	Equipe (Discord)	1h	1h
Repositories e models com a relação entre entidades e testes	Leandro	4hs	3hs
Criação de services, category e product e	Albert	4hs	4hs

testes inicial			
Criação de DTOs e Mappers	Laio	6hs	5hs
Controller de Product, Category, Cart e testes	Carlos	24hs	24hs
Flyway (Migrations) Inicializada	Albert	2hs	01:30hs
Refatoração de repository e models	Laio	6hs	6hs
Modulo do security	Albert e Leandro	5hs	4:30hs
Refatoração de services	Laio	5hs	5hs
Refatoração do security e envio de email	Albert	8hs	8hs
Refatoração dos DTOs services, mappers e controllers	Laio	6hs	6hs
Controller de Usuario	Albert e Laio	2hs	2hs
Permisões de ADMIN E CLIENT	Albert		
Teste de entities	Leandro	2hs	2hs
Teste dos repositories e services	Leandro	24hs	24hs
Teste do service de UserSystem	Biancardy	24hs	
Teste dos controllers	Carlos	24hs	24hs
Migrations	Daniel	3hs	03hs
Controller de Payment e service	Laio	5hs	5hs
Migrations refatoração	Albert	1h	1h
Documentação, swagger	Albert e Carlos	24hs	18hs
Slide da apresentação	Albert e Leandro	4hs	4hs
Implementação do score do product	Laio	3hs	3hs

RELATÓRIO DE ATIVIDADE: OBSERVAÇÕES

Reunião Inicial

No dia 14 de maio de 2025, às 18h, realizamos uma reunião com todos os membros da equipe para alinhar as ações referentes ao projeto. As principais pautas discutidas incluíram a modelagem do banco de dados, que foi concluída durante a reunião, e a divisão de tarefas. Foram definidos os seguintes responsáveis: Leandro ficou responsável pelos models e repositories, Albert pelos services, Rlaio pelos DTOs e Carlos pelos controllers. Além disso, agendamos um novo encontro para a próxima sexta-feira, dia 17 de maio, para a apresentação das tarefas concluídas.

Avaliação do Progresso

No dia 17 de maio de 2025, às 18h, realizamos a segunda reunião com a equipe para avaliar o progresso das atividades que deveriam ser entregues naquele dia. Durante a reunião, percebemos a necessidade de padronizar a linguagem do código em inglês, pois havia uma mistura de palavras entre os dois idiomas. Também discutimos e definimos como os DTOs e Mappers seriam padronizados. Decidimos refatorar o código para essas funcionalidades. Nem todos os membros do grupo puderam participar devido ao horário. Entre os dias 14 e 17, realizamos reuniões pontuais com alguns participantes da equipe para garantir o alinhamento e o progresso das atividades.

Avaliação de Pendencias

No dia 20, foi realizada uma reunião para analisar as pendências e resolvê-las, permitindo assim o desenvolvimento contínuo e tranquilo do projeto. Percebemos que era necessário refatorar os testes devido às alterações nos DTOs, models, services, repositories e controllers. Essas atividades de testes foram designadas aos respectivos responsáveis. Além disso, era necessário realizar as migrations, pois precisavam ser atualizadas.

Verificação de Endpoints

Na reunião do dia 23, a equipe se reuniu para verificar as requisições feitas pelo Postman e assegurar que estavam funcionando conforme o esperado. Durante a verificação, identificamos alguns problemas nos retornos de erro. Embora simples de resolver, devido a outras demandas prioritárias, decidimos adiar a correção desses problemas.

Decisão Técnica

Optamos por não resolver imediatamente esses problemas menores, pois o foco era garantir uma entrega do aplicativo funcional e segura dentro do prazo estabelecido. Priorizar funcionalidades críticas e segurança

foi essencial para atender às expectativas do cliente e assegurar a robustez do sistema.

Plano de Ação

Para abordar os problemas identificados, adotaremos a seguinte abordagem:

1. **Versionamento:** Criar uma nova versão do sistema para resolver esses problemas de retorno de erro.
2. **Atualizações Posteriores:** Planejar uma atualização do sistema para corrigir esses detalhes, garantindo que a aplicação continue evoluindo de forma organizada e sem comprometer a funcionalidade atual.

Conclusão

A decisão de postergar a correção dos problemas de retorno de erro foi tomada com base na priorização das demandas mais críticas, garantindo a entrega de um aplicativo funcional e seguro. A criação de uma nova versão para tratar esses problemas posteriormente assegura que o sistema continuará a ser aprimorado de maneira estruturada e eficiente.

Através das reuniões e alinhamentos pontuais, conseguimos identificar e corrigir problemas de padronização, garantindo que todos os membros da equipe estivessem cientes de suas responsabilidades. Esse processo promoveu um desenvolvimento mais coeso e organizado, permitindo-nos focar nas funcionalidades essenciais e na segurança, enquanto mantemos um plano claro para resolver os detalhes menores no futuro. Com essa abordagem, asseguramos a entrega de um produto de alta qualidade e preparado para futuras atualizações.

CONCLUSÃO SOBRE O PROCESSO DE DESENVOLVIMENTO

No processo de atualização do software, realizamos uma série de melhorias e implementações para aumentar a escalabilidade e a manutenibilidade do sistema. Foram feitas reorganizações de pacotes, implementação de padrões DTO com FORM e VIEW, uso da biblioteca Mapper, padronização de exceções, configuração de fuso horário, documentação da API, validações, uso de migrations, testes com Mockito, e melhorias de segurança com Spring Security e JWT. Além disso, o envio de e-mails personalizados utilizando Thymeleaf foi adicionado.

Nosso processo de desenvolvimento começou com a modelagem do SGBD, seguida pelo mapeamento das entidades relacionadas e a implementação dos repositórios, juntamente com seus testes. Em seguida, desenvolvemos os DTOs e Mappers, e posteriormente os serviços e seus testes. Por fim, implementamos os controladores.

Durante esse processo, enfrentamos alguns imprevistos, como a mistura de escrita em português e inglês, o que exigiu a padronização para uma única linguagem, sendo escolhido o inglês. Também tomamos decisões importantes sobre como implementar o padrão DTO em relação à nomenclatura e às diferentes abordagens, além de determinar as maneiras de utilizar a biblioteca Mapper.

O maior desafio foi tomar decisões sem experiência prévia em liderança de equipe, considerando as diferentes características e vidas dos desenvolvedores. Conciliar horários de reuniões e prazos de entrega das atividades foi complicado, especialmente porque nem todos possuíam as habilidades necessárias para certas tarefas. Diante disso, foi necessário traçar um plano para organizar as tarefas de forma eficiente. Enquanto um membro do grupo refatorava os DTOs e Mappers e padronizava o código em inglês, para atualizar a branch develop, os outros componentes da equipe continuavam executando suas tarefas, nas suas respectivas branch.

Atualizar o software dentro do prazo estipulado foi desafiador, pois o prazo era rígido e não permitia perder muito tempo em pesquisas para melhorias de desempenho. Apesar desses desafios, as soluções implementadas melhoraram significativamente a escalabilidade e manutenibilidade do software. A reorganização dos pacotes, implementação de padrões DTO com FORM e VIEW, uso da biblioteca Mapper, padronização de exceções com a classe ApiExceptionHandler, configuração de fuso horário com a classe SpringTimezoneConfig, documentação da API com Swagger, validações precisas, uso de migrations, testes com Mockito, segurança com a versão mais recente do Spring Security e JWT, e o envio de e-mails personalizados utilizando Thymeleaf contribuíram para um sistema mais robusto e eficiente.