# Sumário do Resultado - Design e Implementação do Software de Reserva de Salas

**Autor:** Henrique Bezerra de Jesus **Data:** 17/08/2024

## ATIVIDADE 1.1

### Passo 1: Identificação das fases do SDLC

As principais fases do SDLC aplicadas ao desenvolvimento do programa de reserva de salas são:

- **Levantamento de Requisitos:** Coletar as necessidades dos usuários e stakeholders sobre o sistema de reservas, incluindo autenticação, gerenciamento de privilégios, e funcionalidades de reserva e gerenciamento de conflitos.
- **Análise:** Analisar os requisitos coletados para definir os requisitos funcionais e não funcionais, como desempenho e segurança.
- **Design:** Criar o design do sistema, incluindo a arquitetura geral, design de interface do usuário (UI) e a estrutura do banco de dados.
- **Implementação (Codificação):** Escrever o código-fonte com base nos designs especificados, utilizando linguagens adequadas.
- **Testes:** Verificar se o software funciona conforme esperado através de testes unitários, de integração, de sistema e de aceitação.
- **Implantação:** Instalar o sistema no ambiente de produção, configurar servidores e bancos de dados, e treinar usuários.
- **Manutenção:** Fornecer suporte contínuo, incluindo correção de bugs e melhorias.

### Passo 2: Definição dos papéis e responsabilidades

- **Gerente de Projeto:** Coordenar todas as fases do SDLC, gerenciar prazos e recursos.
- **Analista de Requisitos:** Coletar e documentar requisitos funcionais e não funcionais.
- **Arquiteto de Sistemas:** Criar o design arquitetural do sistema.
- **Desenvolvedores:** Implementar o código conforme o design e realizar testes unitários.
- **Testadores/Engenheiros de QA:** Executar planos de teste e identificar bugs.
- **Engenheiro de DevOps:** Gerenciar infraestrutura e processos de implantação.
- **Engenheiro de Suporte/Manutenção:** Fornecer suporte técnico e resolver problemas.

### Passo 3: Levantamento de Requisitos do Software

**Requisitos Funcionais:**

- **Autenticação de Usuários:** Login com privilégios diferenciados.
- **Reserva de Sala:** Reserva de salas para horários específicos.
- **Listagem de Reservas:** Listar todas as reservas existentes.
- **Cancelamento de Reservas:** Apenas admin pode cancelar reservas.
- **Gerenciamento de Conflitos:** Prevenir reservas conflitantes.

**Requisitos Não Funcionais:**

- **Segurança:** Proteção de dados com criptografia.
- **Desempenho:** Responder a solicitações em menos de 2 segundos.
- **Usabilidade:** Interface intuitiva e acessível.

### Passo 4: Design e Implementação do Software

**Esboço do Design da Interface do Usuário:**

- **Tela de Login:** Campos para nome de usuário, senha, e botões para login e recuperação de senha.
- **Tela de Reserva:** Formulário para selecionar sala, data e hora.
- **Tela de Listagem de Reservas:** Lista de reservas com opções para visualizar detalhes e cancelar.

**Principais Passos para Implementação:**

- **Configuração do Ambiente de Desenvolvimento:** Preparar ferramentas necessárias.
- **Criação do Banco de Dados:** Definir tabelas e relações.
- **Desenvolvimento do Backend:** Implementar lógica de negócios e autenticação.
- **Desenvolvimento do Frontend:** Implementar a interface do usuário.
- **Integração e Testes Unitários:** Integrar frontend e backend e realizar testes.

### Passo 5: Testes do Software

**Plano de Testes:**

- **Teste de Autenticação:** Verificar login e gerenciamento de privilégios.
- **Teste de Reserva:** Validar criação de reservas e prevenção de conflitos.
- **Teste de Cancelamento:** Garantir que apenas admins possam cancelar reservas.
- **Teste de Usabilidade:** Avaliar a facilidade de uso da interface.

## ATIVIDADE 1.2 - Planejamento Ágil e User Stories

### Passo 1: Definição do Product Backlog

**Conversão para User Stories:**

1. **Como um usuário**, eu quero fazer login no sistema para acessar minhas reservas.
    - **Critérios de Aceitação:** A tela de login deve permitir que o usuário entre com um nome de usuário e senha válidos. Após o login, o usuário deve ser redirecionado para a página de reservas.
2. **Como um admin**, eu quero cancelar uma reserva para gerenciar a disponibilidade de salas.
    - **Critérios de Aceitação:** A tela de listagem de reservas deve exibir um botão de cancelamento ao lado de cada reserva. Somente usuários com privilégios de admin devem conseguir cancelar reservas.
3. **Como um usuário**, eu quero reservar uma sala para um horário específico para garantir seu uso.
    - **Critérios de Aceitação:** A tela de reserva deve permitir a seleção de uma sala, data e horário. Após a confirmação, a reserva deve ser registrada no sistema e a disponibilidade da sala deve ser atualizada.

**Priorização das User Stories:**

4. **Login do Usuário** - Alta prioridade: Essencial para acesso ao sistema.
5. **Reserva de Sala** - Alta prioridade: Funcionalidade central do sistema.
6. **Cancelamento de Reserva (Admin)** - Média prioridade: Importante para o gerenciamento de reservas, mas acessível apenas para admins.

**Produto Final do Passo 1:**

- Product Backlog com as User Stories priorizadas e critérios de aceitação definidos.

### Passo 2: Planejamento de Sprints

**Seleção das Funcionalidades:**

Para o primeiro Sprint, selecionamos as User Stories mais prioritárias:

- Login do Usuário
- Reserva de Sala

**Divisão em Tarefas Menores:**

7. **User Story: Login do Usuário**
    - Criar tela de login.
    - Implementar autenticação.
    - Testar funcionalidades de login.

8. **User Story: Reserva de Sala**
    - Criar tela de reserva.
    - Implementar lógica de reserva e validação.
    - Testar funcionalidades de reserva.

**Criação do Plano de Sprint:**

- **Sprint 1:**
    - **Semana 1:** Desenvolvimento da tela de login e autenticação.
    - **Semana 2:** Desenvolvimento da tela de reserva e lógica de reserva.
    - **Semana 3:** Testes das funcionalidades de login e reserva.

**Produto Final do Passo 2:**

- Plano de Sprint detalhado com tarefas, estimativas e cronograma.

**Passo 3: Desenvolvimento Iterativo**

**Execução das Tarefas:**

- Implementar as funcionalidades conforme o plano de Sprint.
- Realizar revisões contínuas e ajustar conforme necessário.

**Colaboração e Feedback:**

- Realizar reuniões diárias para verificar o progresso.
- Buscar feedback dos stakeholders e ajustar o desenvolvimento conforme necessário.

**Produto Final do Passo 3:**

- Funcionalidades implementadas e feedback incorporado.

**Passo 4: Revisão e Feedback**

**Revisão do Trabalho Realizado:**

- Revisar as funcionalidades implementadas (login e reserva).
- Garantir que atendem às expectativas dos usuários.

**Coleta de Feedback:**

- Solicitar feedback dos usuários e stakeholders sobre a funcionalidade de login e reserva.
- Registrar observações e melhorias sugeridas.

**Produto Final do Passo 4:**

- Registro das observações e feedback recebido.

**Passo 5: Retrospectiva e Planejamento da Próxima Iteração**

**Retrospectiva:**

- Avaliar o que funcionou bem e o que pode ser melhorado no processo de desenvolvimento.
- Identificar práticas que precisam ser ajustadas.

**Planejamento da Próxima Iteração:**

- Planejar a implementação das User Stories restantes e ajuste no backlog com base nas lições aprendidas.

**Produto Final do Passo 5:**

- Plano ajustado para a próxima iteração com melhorias incorporadas.

## ATIVIDADE 1.3 - Ferramentas DevOps e Infraestrutura

**Passo 1: Escolha das Ferramentas DevOps Open Source**

**Controle de Versão:**

- **Ferramenta Selecionada:** Git
- **Motivo da Escolha:** Git é amplamente adotado e ideal para controle de versão distribuído, facilitando a colaboração em equipe.

**Integração Contínua:**

- **Ferramenta Selecionada:** Jenkins
- **Motivo da Escolha:** Jenkins é altamente customizável e suporta uma ampla gama de plugins para integração contínua.

**Automação:**

- **Ferramenta Selecionada:** Ansible
- **Motivo da Escolha:** Ansible simplifica a automação de configuração e gerenciamento de servidores com uma abordagem baseada em YAML.

**Orquestração de Containers:**

- **Ferramenta Selecionada:** Kubernetes
- **Motivo da Escolha:** Kubernetes oferece suporte robusto para escalabilidade e gerenciamento de containers, com uma grande comunidade de suporte.

**Monitoramento:**

- **Ferramenta Selecionada:** Prometheus + Grafana

- **Motivo da Escolha:** Prometheus coleta métricas e Grafana oferece uma interface para visualização, ambas amplamente suportadas.

**Observações:**

- **Positivas:** Ferramentas bem suportadas pela comunidade, facilitando a resolução de problemas.
- **Negativas:** Complexidade inicial de configuração pode ser um desafio.

**Passo 2: Planejamento da Infraestrutura**

**Servidores:**

- Utilização de **AWS EC2** para flexibilidade e escalabilidade.
- **Containers Docker** para gerenciamento simplificado.

**Bancos de Dados:**

- **Ferramenta Selecionada:** PostgreSQL
- **Motivo da Escolha:** PostgreSQL é robusto e open source, com suporte a transações e consistência.

**Segurança:**

- **Implementação:** Firewalls e controle de acesso com regras baseadas em segurança.
- **Ferramenta:** Vault para gestão de segredos.

**Backups:**

- **Configuração:** Backups automatizados do banco de dados e arquivos críticos usando **AWS S3**.

**Observações:**

- **Positivas:** Escalabilidade e flexibilidade da infraestrutura em nuvem.
- **Negativas:** Custos crescentes com a expansão da base de usuários.

**Passo 3: Organização do Ciclo de Desenvolvimento**

**Fluxo de Trabalho:**

- **Pipelines CI/CD:** Jenkins para automação de build, testes e deploy.
- **Estratégia de Branching:** GitFlow para organização do desenvolvimento.

**Automação de Testes:**

- **Configuração:** Testes automatizados com **Selenium** (interface) e **JUnit** (unidade).

**Entrega Contínua:**

- **Deployment Automatizado:** Usar Ansible e Kubernetes, com validação e rollback.

**Observações:**

- **Positivas:** Automação reduz erros manuais e aumenta agilidade.
- **Negativas:** Curva de aprendizado pode ser alta para novas ferramentas.

**Passo 4: Elaboração do Cronograma e Alocação de Recursos**

**Cronograma:**

- **Divisão:** Sprints de 2 semanas com marcos para funcionalidades críticas, testes e implantação.

**Alocação de Recursos:**

- **Equipe:** Desenvolvedores, DevOps, QA.
- **Treinamento:** Tempo alocado para treinamento nas ferramentas DevOps.

**Observações:**

- **Positivas:** Sprints permitem ajustes rápidos com base em feedback.
- **Negativas:** Subestimação de tempos pode levar a atrasos.

**Passo 5: Plano de Segurança e Escalabilidade**

**Segurança:**

- **Implementação:** Autenticação via OAuth2, criptografia com TLS, e políticas de acesso rigorosas.
- **Ferramenta:** Vault para senhas e chaves de criptografia.

**Escalabilidade:**

- **Planejamento:** Escalar horizontalmente com Kubernetes, usando autoscaling.

**Observações:**

- **Positivas:** Segurança robusta e escalabilidade garantem suporte ao crescimento.
- **Negativas:** Complexidade adicional pode exigir monitoramento contínuo.

## ATIVIDADE 1.4 - Análise de Ameaças e Riscos

**Passo 1: Análise de Ameaças e Riscos**

**Identificação de Ameaças:**

9. **SQL Injection:**
    - **Descrição:** Manipulação de consultas SQL através de entradas maliciosas.
    - **Gravidade:** Alta
    - **Probabilidade:** Média
10. **Cross-Site Scripting (XSS):**
    - **Descrição:** Injeção de scripts maliciosos em páginas web.
    - **Gravidade:** Média
    - **Probabilidade:** Alta
11. **Ataque de Força Bruta:**
    - **Descrição:** Tentativas automatizadas de adivinhar credenciais de login.
    - **Gravidade:** Alta
    - **Probabilidade:** Baixa

**Observações:**

- **Positivas:** Análise proporciona uma base sólida para priorizar segurança.
- **Negativas:** Avaliação subjetiva pode necessitar revisões contínuas.

**Passo 2: Definição de Controles de Segurança (continuação)**

**Controle para XSS:**

- **Implementação:**
    - **Validação de Entrada:** Filtrar e validar entradas do usuário para prevenir a inclusão de scripts maliciosos.
    - **Escape de Saída:** Escapar caracteres especiais em saídas para garantir que o código HTML seja tratado como texto, não como código executável.

**Controle para Ataque de Força Bruta:**

- **Implementação:**
    - **Bloqueio de IP:** Implementar mecanismos de bloqueio temporário após um número específico de tentativas de login falhadas.
    - **CAPTCHA:** Adicionar CAPTCHA para dificultar a automação dos ataques.

**Passo 3: Avaliação de Controles e Procedimentos**

**Avaliação da Eficácia dos Controles:**

- **SQL Injection:**
    - **Testes:** Realizar testes de penetração e auditoria de código para verificar a eficácia das consultas parametrizadas e do ORM.

    - **Revisões:** Revisar as práticas de codificação e garantir que todos os pontos de entrada do banco de dados usem consultas seguras.
- **XSS:**
    - **Testes:** Utilizar ferramentas de análise de segurança para verificar se há vulnerabilidades de XSS e realizar testes manuais.
    - **Revisões:** Revisar o código para garantir que todas as saídas sejam devidamente escapadas e que a validação de entrada esteja em vigor.
- **Força Bruta:**
    - **Testes:** Simular ataques de força bruta para verificar se as medidas de bloqueio e CAPTCHA funcionam conforme esperado.
    - **Revisões:** Monitorar logs de acesso e ajustar os limites de bloqueio conforme necessário para balancear a segurança e a usabilidade.

**Observações:**

- **Positivas:** Implementação eficaz dos controles melhora a segurança geral do sistema.
- **Negativas:** Necessidade de revisão contínua e ajuste dos controles para acompanhar novas ameaças.

**Passo 4: Plano de Monitoramento e Resposta**

**Monitoramento Contínuo:**

- **Implementação:** Utilizar ferramentas de monitoramento (como Prometheus e Grafana) para observar logs e atividades suspeitas.
- **Alertas:** Configurar alertas para atividades anômalas que possam indicar uma tentativa de ataque ou brecha de segurança.

**Resposta a Incidentes:**

- **Plano de Resposta:** Criar um plano de resposta a incidentes que inclua procedimentos para identificação, contenção, erradicação e recuperação de ataques.
- **Treinamento:** Treinar a equipe para reagir rapidamente a incidentes de segurança e realizar simulações de ataques.

**Observações:**

- **Positivas:** Monitoramento contínuo e resposta a incidentes garantem que o sistema possa responder rapidamente a ameaças.
- **Negativas:** Monitoramento intensivo pode gerar um grande volume de dados e requer recursos para análise.

**Passo 5: Documentação e Atualização**

**Documentação:**

- **Registro:** Manter um registro detalhado das ameaças identificadas, controles implementados e avaliações realizadas.
- **Atualização:** Atualizar a documentação regularmente para refletir mudanças na infraestrutura e novas ameaças identificadas.

**Observações:**

- **Positivas:** Documentação completa e atualizada ajuda na manutenção da segurança e na conformidade com as melhores práticas.
- **Negativas:** Manter a documentação atualizada pode ser trabalhoso e exigir revisões frequentes.

## ATIVIDADE 1.5 - Configuração e Gerenciamento de Ambiente

**Passo 1: Configuração do Ambiente de Desenvolvimento**

**Ferramentas e Tecnologias:**

- **Configuração:**
  - **IDE:** Configurar IDEs como PyCharm ou VSCode com plugins necessários para desenvolvimento em Flask e SQLite.
  - **Ambiente Virtual:** Utilizar `venv` para criar ambientes virtuais e gerenciar dependências.
  - **Versão do Python:** Garantir que a versão do Python seja compatível com as bibliotecas utilizadas.

**Passo 2: Configuração do Ambiente de Testes**

**Testes Automatizados:**

- **Ferramentas:**
  - **Unittest ou Pytest:** Configurar frameworks de teste para criar e executar testes automatizados.
  - **Integração Contínua:** Configurar Jenkins para executar testes automatizados em cada commit.

**Ambiente de Testes:**

- **Isolamento:** Configurar um ambiente de testes separado para garantir que as mudanças não afetem o ambiente de produção.

**Passo 3: Configuração do Ambiente de Produção**

**Infraestrutura:**

- **Servidores:**
  - **Configuração:** Configurar servidores (AWS EC2) com as especificações necessárias para suportar a aplicação.

    - **Containers:** Utilizar Docker para empacotar a aplicação e Kubernetes para orquestrar os containers.

**Bancos de Dados:**

- **Configuração:** Configurar PostgreSQL e garantir que esteja otimizado para o ambiente de produção.
- **Backup:** Configurar backups automáticos e estratégias de recuperação de desastres.

**Segurança:**

- **Implementação:** Configurar firewalls e políticas de segurança para proteger o ambiente de produção.
- **Monitoramento:** Implementar ferramentas de monitoramento para observar o desempenho e a segurança do sistema.

**Passo 4: Documentação e Procedimentos**

**Documentação:**

- **Registro:** Documentar a configuração de todos os ambientes e os procedimentos para implantação e manutenção.
- **Atualização:** Manter a documentação atualizada com quaisquer mudanças na infraestrutura ou na configuração.

**Procedimentos:**

- **Deploy:** Estabelecer procedimentos claros para o deploy de atualizações e mudanças.
- **Manutenção:** Criar um plano de manutenção regular para garantir que o sistema permaneça seguro e eficiente.

**Observações:**

- **Positivas:** A configuração adequada dos ambientes garante que o sistema funcione de forma eficiente e segura em produção.
- **Negativas:** A gestão e manutenção dos ambientes podem ser complexas e requerem atenção contínua.