ANO
2025

UNINTER

PROJETO MULTIDISCIPLINAR

ORIENTAÇÕES PARA O DESENVOLVIMENTO DOS PROJETOS

Prof. Winston Sen Lun Fung, Me.

INTRODUÇÃO

Este documento reúne instruções detalhadas para orientar o desenvolvimento individual do **Projeto “SGHSS” (Sistema de Gestão Hospitalar e de Serviços de Saúde)**. A proposta está organizada em fases semanais, contemplando desde o planejamento inicial até a entrega final, incorporando exemplos e boas práticas.

SUMÁRIO

FASE 1 (SEMANA 1): PLANEJAMENTO E COMPREENSÃO

1. Leitura Aprofundada do Estudo de Caso

- **Objetivo**: Entender o cenário da instituição VidaPlus, os requisitos (funcionais e não funcionais) e os principais atores (pacientes, profissionais de saúde, administradores).
- **Ações**:
    - Identifique problemas que o sistema resolve (agendamentos, prontuários, telemedicina, etc.).
    - Destaque pontos críticos, como segurança (LGPD) e escalabilidade.

2. Definição do Escopo Individual

- **Escolha a Ênfase**:
    - **Back-end** (modelagem de dados, API, regras de negócio);
    - **Front-end** (interfaces, responsividade, usabilidade);
    - **Qualidade de Software** (planos de teste, testes funcionais, não funcionais, segurança).
- **Estabeleça Prioridades**: selecione as funcionalidades mais relevantes para demonstrar suas habilidades dentro do tempo disponível.

3. Cronograma e Organização

- **Monte um cronograma** de 8 semanas (ou conforme seu calendário). Por exemplo:
    - **Semana 1**: Planejamento e compreensão;
    - **Semanas 2 - 3**: Modelagem e arquitetura;
    - **Semanas 4 - 6**: Implementação (ou prototipagem);
    - **Semana 7**: Plano de testes e qualidade;
    - **Semana 8**: Documentação final e revisão.
- **Ferramentas de apoio**: Use planilhas, aplicativos de tarefas (Trello, Asana) ou simplesmente um cronograma no Word para se organizar.

4. Pesquisa de Referências e Ferramentas

- **Back-end**: Pesquise frameworks (Node.js, Python/Django, Java/Spring Boot), padrões de projeto (MVC, DDD), bancos de dados (MySQL, PostgreSQL entre outros).
- **Front-end**: Explore frameworks (React, Angular, Vue.js), bibliotecas de componentes (Bootstrap, Material UI) e ferramentas de design (Figma, Adobe XD).

- **Qualidade**: Investigue metodologias (TDD, BDD), ferramentas de automação (Selenium, Cypress), ferramentas de carga (JMeter, Locust) e de segurança (OWASP ZAP).

FASE 2 (SEMANAS 2 - 3): MODELAGEM E ARQUITETURA

1. Análise de Requisitos Detalhada

- **Expanda** os requisitos funcionais e não funcionais, descrevendo o que o sistema deve fazer e quais critérios de desempenho, segurança, usabilidade e conformidade serão adotados.
- **Tabela de Requisitos** (Exemplo):

ID	Descrição	Tipo	Prioridade
RF001	Permitir cadastro de pacientes (dados pessoais, clínicos)	Funcional	Alta
RF002	Agendar consultas online	Funcional	Alta
RNF001	Interface responsiva em dispositivos móveis	Não Funcional	Média

2. Diagramas UML e Modelos

- **Casos de Uso**: Mostre como pacientes, profissionais e administradores interagem com as funcionalidades.
- **Diagrama de Classes (para Back-end)**: Defina classes (Paciente, Médico, Consulta etc.), atributos e métodos, além dos relacionamentos (1:N, N:N).
- **Wireframes / Mockups (para Front-end)**: Esboce telas de login, agendamento, visualização de prontuário.
- **Fluxos de Teste (para Qualidade)**: Use diagramas de atividades ou casos de uso para identificar cenários de teste.

3. Definição da Arquitetura

- **Back-end:** Decidir se usará arquitetura monolítica ou microserviços, se haverá camadas de serviços, repositórios etc.
- **Front-end:** Estabelecer como organizar componentes, rotas, design responsivo, padrões de layout.
- **Qualidade**: Planejar uso de integração contínua, pipelines de teste, cobertura de código, relatórios automatizados.

FASE 3 (SEMANAS 4 - 6): IMPLEMENTAÇÃO (OU PROTOTIPAGEM)

1. Protótipo de Telas (Para Foco em Front-end)

- Crie telas que representem o fluxo principal (login, cadastro de paciente, agendamento de consulta).
- Ferramenta Sugerida: Figma, Adobe XD (para protótipo) ou HTML/CSS/JS + framework (para protótipo funcional).

2. Desenvolvimento do Código (Para Foco em Back-end)

- Modelo de Dados: Crie o esquema do banco (DER) e implemente as entidades.
- API REST: Implemente endpoints para operações de CRUD e funcionalidades (ex.: `POST /pacientes`, `GET /consultas`).
- Boas Práticas: Utilize Git para versionamento, padronize a nomenclatura de classes e métodos, implemente logs e tratamento de erros.

3. Simulação de Implementação (Para Foco em Qualidade)

- Caso não implemente um sistema completo, apresente pseudocódigo ou scripts exemplificando como seriam testadas as principais funcionalidades.
- Exemplo: Pseudocódigo de um método `agendarConsulta()`, listando entradas, processamentos e saídas esperadas.

FASE 4 (SEMANA 7): PLANO DE TESTES E QUALIDADE

1. Casos de Teste

- Defina claramente entrada, comportamento esperado e resultado.
- Exemplo de Caso de Teste:

Caso	Descrição	Resultado Esperado
CT001	Cadastrar paciente com dados válidos	Exibir mensagem "Paciente cadastrado com sucesso."
CT002	Tentar cadastrar paciente sem informar CPF	Exibir mensagem de erro e impedir cadastro

2. Estratégia de Testes

- Funcionais: Verifique cada requisito funcional (cadastro, agendamento etc.).
- Não Funcionais: Teste desempenho (JMeter), carga (Locust), segurança (OWASP ZAP), usabilidade.
- Integração Contínua: Se possível, configure pipelines para automatizar execuções de teste.

3. Ferramentas de Automação (Para Foco em Qualidade)

- Selenium ou Cypress: Testes de interface.
- JMeter ou Locust: Testes de desempenho e carga.
- OWASP ZAP ou Burp Suite: Testes de segurança (SQL Injection, XSS etc.).

FASE 5 (SEMANA 8): DOCUMENTAÇÃO E REVISÃO FINAL

1. **Montagem do Documento Principal**
   - **Estrutura Recomendada**:
     1. **Capa, Folha de Rosto** e **Sumário**
     2. **Introdução** (contexto e objetivo)
     3. **Requisitos** (funcionais e não funcionais)
     4. **Modelagem e Arquitetura** (UML, wireframes, escolha de tecnologias)
     5. **Implementação** (código, pseudocódigo, prints de tela)
     6. **Plano de Testes** (casos, resultados, screenshots de ferramentas)
     7. **Conclusão** (lições aprendidas, desafios, melhorias futuras)
     8. **Referências** (livros, artigos, sites usados)

2. Materiais Suplementares

- Anexe diagramas (classes, casos de uso), prints de protótipos/telas, scripts de teste ou relatórios de ferramentas.
- Mantenha tudo **organizado** e **coeso** com o corpo principal do documento.

3. Revisão e Ajustes Finais

- **Verifique ortografia** e formatação.
- Confirme se **todos os itens** solicitados pelo professor estão atendidos.
- Faça o **merge** final de arquivos (separados ou não) num único PDF (caso exigido).
- **Backup**: guarde o material em lugar seguro (nuvem ou pendrive).

DICAS E BOAS PRÁTICAS

- **Organização de Arquivos:** Utilize pastas e nomes descritivos para diagramas, códigos e documentos.
- **Comunicação:** Em caso de dúvidas, procure o professor ou tutor via fórum ou canais oficiais.
- **Originalidade:** Evite plágio; cada projeto de TI é único. Faça adaptações pessoais ao seu estudo de caso.
- **Aprendizado Contínuo:** Mesmo após concluir o projeto, use o feedback para aprimorar suas habilidades.

EXEMPLO DE CRONOGRAMA RESUMIDO (SUGESTÃO)

Abaixo está uma sugestão para o desenvolvimento das suas atividades na disciplina de Projetos.

Fase	Semana(s)	Principais Tarefas
Fase 1: Planejamento e Compreensão	1	Ler Estudo de Caso, definir ênfase, montar cronograma
Fase 2: Modelagem e Arquitetura	2 - 3	Definir requisitos, criar diagramas UML, planejar arquitetura
Fase 3: Implementação / Prototipagem	4 - 6	Desenvolver protótipo ou código, simular funcionalidade
Fase 4: Testes e Qualidade	7	Planejar e executar testes (funcionais, não funcionais)
Fase 5: Documentação e Revisão Final	8	Finalizar documentação, anexar artefatos, revisar e entregar

CONCLUSÃO

Seguindo este roteiro, você conseguirá conduzir seu projeto individual de forma estruturada, demostrando competências nas áreas de Análise de Sistemas, Engenharia de Software, Desenvolvimento (Back-end ou Front-end) ou Qualidade de Software. A divisão semanal ajuda a organizar o tempo e a mensurar o progresso, garantindo que cada parte essencial do projeto — desde o entendimento dos requisitos até a entrega final — seja abordada com a devida atenção.

Bons estudos e sucesso no seu Projeto SGHSS!