Desafio

Endereço: <https://gmtavora.000webhostapp.com/>

~~Nome de usuário: admin | Senha: admin~~

## Requisitos de Software

Requisitos de software são necessidades ou exigências de um negócio. O sistema pode ser tratado como um escopo dividido em módulos, nos quais são estabelecidos requisitos funcionais e regras de negócios.

Requisitos mal-elaborados estão entre as principais razões para o fracasso de um software. Elaborar bons requisitos requer conhecimento do domínio, uma boa comunicação entre o usuário e o analista, além de boa gestão de evolução e histórico dos requisitos.

Cada requisito é um aspecto do que o sistema proposto deve fazer, ou ainda uma restrição no desenvolvimento do sistema. É importante ressaltar que o sistema deve resolver os problemas do cliente e não simplesmente fazer o que o analista ou o programador deseja. Isso não significa que um analista ou programador não possa contribuir com sugestões e propostas. O conjunto dos requisitos deve ser um contrato entre as partes interessadas no sistema.

Os requisitos de software podem ser classificados em requisitos funcionais, que tratam das funcionalidades e serviços do sistema, requisitos não funcionais, que definem propriedades e restrições do sistema, como tempo, espaço, sistema operacional etc., e ainda em regras de negócio, que são premissas e restrições aplicadas a uma operação comercial da empresa de forma que o negócio ocorra da maneira esperada.

Por fim, os requisitos devem fornecer uma base para estimar o custo e o tempo de desenvolvimento do sistema. O DEP (Documento de Especificação do Projeto) deve realizar o alinhamento entre as expectativas entre o cliente e a empresa contratada.

Referências:

<https://www.ateomomento.com.br/requisitos-de-software/>

<https://www.devmedia.com.br/introducao-a-requisitos-de-software/29580>

<https://fluxoconsultoria.poli.ufrj.br/blog/tecnologia-informacao/requisitos-de-software/>

## Qualidade de Software

Um software de qualidade é fácil de usar, funciona corretamente, é de fácil manutenção e mantém a integridade dos dados para evitar possíveis falhas. Os custos resultantes de defeitos ou erros provocados por falhas de softwares podem ser catastróficos, gerando um impacto, muitas vezes, irremediável.

Os requisitos de software são a base a partir da qual a qualidade é medida. A falta de conformidade aos requisitos significa falta de qualidade. Padrões especificados definem um conjunto de critérios de desenvolvimento que orientam a maneira segundo a qual o software passa pelo trabalho de engenharia. Se os critérios não forem seguidos, o resultado quase que seguramente será a falta de qualidade. Além disso, existe um conjunto de requisitos implícitos que frequentemente não são mencionados na especificação (por exemplo, o desejo de uma boa manutenibilidade).

Para se garantir a qualidade de um produto durante seu desenvolvimento existe um conjunto de métodos e técnicas que devem ser implementadas durante todo o processo. Entre eles:

* Identificação de defeitos: quanto antes a presença do defeito for revelada, menor o custo de correção do defeito e maior a probabilidade de corrigi-lo corretamente. A solução é introduzir atividades de Validação, Verificação e Testes (VV&T) ao longo de todo o ciclo de desenvolvimento:

- VV&T Estática: Revisões de Software (Revisões aos Pares, Inspeções de Software, entre outros). Denomina-se estática, pois não envolve a execução do produto, mas apenas leitura de artefatos, como código.

- VV&T Dinâmica: Testes de Software (Testes de unidade, de integração, do software, entre outros).

* Revisões de Software: São processos ou atividades para leitura de um artefato de software visando assegurar que ele cumpre sua especificação e atende às necessidades de seus usuários.
* Testes de Software: Processo de executar um programa ou sistema com o objetivo de revelar a presença de erros; ou, falhando nesse objetivo, aumentar a confiança sobre o programa.

Para guiar os analistas a identificar o real propósito das informações obtidas junto ao usuário, existe um sistema de classificação de requisitos chamado FURPS+, detalhado abaixo:

* Funcionalidade: é todo o aspecto funcional do sistema, que deve satisfazer as necessidades explícitas e implícitas para a finalidade a que se destina o produto.
* Usabilidade: indica o tempo de treinamento necessário para o usuário se tornar produtivo, além de todo o material de ajuda e documentação do usuário;
* Confiabilidade: refere-se à precisão, número máximo de defeitos, impacto das falhas, etc.;
* Desempenho: indica o tempo de resposta para uma transação, capacidade, operação parcial e uso de recursos;
* Suportabilidade: define o padrão de codificação, convenção de nomenclatura, bibliotecas de classes, utilitários de manutenção;
* +: refere-se ao design, implementação, interface etc.

Referências:

<https://www.devmedia.com.br/qualidade-de-software/9408>

<http://www.linhadecodigo.com.br/artigo/1712/qualidade-qualidade-de-software-e-garantia-da-qualidade-de-software-sao-as-mesmas-coisas.aspx>

<https://edisciplinas.usp.br/pluginfile.php/57546/mod_resource/content/1/Aula8-QualidadeSoftware.pdf>

## Scrum

Scrum é uma metodologia ágil para gestão e planejamento de projetos de software.

No Scrum, os projetos são divididos em ciclos (tipicamente mensais) chamados de Sprints. O Sprint representa um Time Box dentro do qual um conjunto de atividades deve ser executado. Metodologias ágeis de desenvolvimento de software são iterativas, ou seja, o trabalho é dividido em iterações, que são chamadas de Sprints no caso do Scrum.

As funcionalidades a serem implementadas em um projeto são mantidas em uma lista que é conhecida como Product Backlog. No início de cada Sprint, faz-se um Sprint Planning Meeting, ou seja, uma reunião de planejamento na qual o Product Owner prioriza os itens do Product Backlog e a equipe seleciona as atividades que ela será capaz de implementar durante o Sprint que se inicia. As tarefas alocadas em um Sprint são transferidas do Product Backlog para o Sprint Backlog.

A cada dia de uma Sprint, a equipe faz uma breve reunião (normalmente de manhã), chamada Daily Scrum. O objetivo é disseminar conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho do dia que se inicia.

Ao final de um Sprint, a equipe apresenta as funcionalidades implementadas em uma Sprint Review Meeting. Finalmente, faz-se uma Sprint Retrospective e a equipe parte para o planejamento do próximo Sprint. Assim reinicia-se o ciclo.

<https://www.desenvolvimentoagil.com.br/scrum/>

## Desenvolvimento orientado por testes (TDD)

A ideia de TDD é bem simples: escreva seus testes antes mesmo de escrever o código de produção. A prática de TDD agrega muitos benefícios ao processo de desenvolvimento. O primeiro deles, e mais claro, são os benefícios na qualidade externa do produto. A bateria de testes automatizados gerados pela prática dão mais segurança ao desenvolvedor na hora de mudanças.

Os testes automatizados, que rodam em questão de segundos, são executados o tempo todo pelo desenvolvedor. Caso algo pare de funcionar, o desenvolvedor é rapidamente notificado, e consegue corrigir o problema antes de mandar a versão para o cliente.

Além disso, caso o desenvolvedor saiba ler o código de testes com atenção, esse mesmo código pode dar informações importantes sobre a qualidade do código que está sendo produzido. Se está difícil escrever um teste automatizado, é porque provavelmente o código de produção está complicado.

Contudo, não é necessário praticar TDD todo o tempo. Se estamos, por exemplo, escrevendo a implementação de um DAO (classe que se comunica com o banco de dados), talvez escrever os testes antes não vá ajudar tanto, afinal não há grandes decisões de design a serem tomadas em classes como essa, e a funcionalidade tende a ser simples. Escrever o teste depois, portanto, não será tão diferente de escrever o teste antes.

Se pararmos para analisar, o grande responsável pelo aumento da qualidade interna e externa não é o TDD, mas sim o teste automatizado, produzido perante o uso da prática. A pergunta comum é justamente então: Qual a diferença entre fazer TDD e escrever o teste depois?

A diferença é justamente na quantidade de feedback. Quando o desenvolvedor escreve os testes somente ao acabar a implementação do código de produção, ele passou muito tempo sem retorno. Afinal, escrever o código de produção leva tempo. Ao praticar TDD, o desenvolvedor divide seu trabalho em pequenas etapas. Ele escreve um pequeno teste, e implementa um pedaço da funcionalidade. E repete. A cada teste escrito, o desenvolvedor ganha feedback.

Quanto mais cedo o desenvolvedor receber feedback, melhor. Quando se tem muito código já escrito, mudanças podem ser trabalhosas e custar caro. Ao contrário, quanto menos código escrito, menor será o custo de mudança. E é justamente isso que acontece com praticantes de TDD: eles recebem feedback no momento em que mudar ainda é barato.

### Como Funciona?

A mecânica da prática é simples: escreva um teste que falha, faça-o passar da maneira mais simples possível e, por fim, refatore o código. Esse ciclo é conhecido como Ciclo Vermelho-Verde-Refatora

Sempre que um desenvolvedor pega uma funcionalidade para fazer, ele a quebra (muitas vezes mentalmente) em pequenas tarefas. Tarefas essas que exigem a escrita de código. Classes são criadas, outras são modificadas. Ao praticar TDD, o desenvolvedor, antes de começar a fazer essas modificações, explicita esses objetivos. Só que ele faz isso por meio de testes automatizados. O teste em código nada mais é do que um trecho de código que deixa claro o que determinado trecho de código deve fazer.

Ao formalizar esse objetivo na forma de um teste automatizado, esse teste falha, claro; afinal, a funcionalidade ainda não foi implementada. O desenvolvedor então trabalha para fazer esse teste passar.  
 Assim que o teste passar, o desenvolvedor então parte para uma próxima etapa no ciclo, importantíssima para aqueles que tem o sonho de produzir código de qualidade: a hora da refatoração. Refatorar é melhorar o código que já está escrito. A cabeça do desenvolvedor é complicada: quando ele está focado em implementar a funcionalidade, ele raramente está pensando também em qualidade de código. Não tem jeito, é assim que funcionamos. E justamente por isso que, após a implementação da funcionalidade, o desenvolvedor para e melhora a qualidade do código (que já funciona e atende ao requisito do negócio).

<https://tdd.caelum.com.br/>

## Sistemas similares

No site ingresso.com, não é possível realizar uma busca que combine informações como a classificação etária do filme e em quais bairros ele está sendo exibido. O sistema deverá ser capaz de filtrar resultados com uma ou mais restrições.

# Proposta

## Requisitos

* O sistema deve ser implementado na Web
* O usuário pode listar todas as fichas médicas (essencial);
* O usuário pode realizar uma busca entre as fichas (essencial);
* O administrador pode cadastrar novos usuários;
* O administrador pode buscar funcionários;
* O administrador pode excluir funcionários;
* O usuário pode alterar sua própria senha e dados pessoais;

## Versão 1

Todas as páginas possuem um cabeçalho com o nome do sistema e um menu que navega entre as funcionalidades Página inicial, Fichas, Opções e Sair.

Os atributos existentes no arquivo de pacientes são salvos no arquivo de configurações junto com seus rótulos. Os rótulos iniciais são, por padrão, os próprios atributos. Os atributos são sempre exibidos para o usuário através de seus rótulos.

**Login**

* Esta funcionalidade está desativada durante o período de desenvolvimento;

**Painel do usuário (página Inicial)**

* Exibe uma mensagem “Bem-vindo, [Nome do usuário]” e a data de hoje;
* Na primeira seção, exibe um alerta se houver dados não normalizados;
* Na segunda seção, mostra um link para a listagem de pacientes;

**Listar fichas médicas**

* Imprime na tela uma tabela com as informações essenciais dos pacientes;
* O usuário pode controlar o número de registros por página através de uma caixa de texto (apenas entradas numéricas) posicionada acima da tabela: “Exibir # registros por página;
* Botão “Adicionar/remover colunas” (acima da tabela) que ao ser clicado abre um pop-up no qual o usuário pode alterar as informações essenciais exibidas na tabela;
* O cabeçalho de cada coluna pode ser clicado para reordenar a listagem. Um ícone é exibido indicando se a ordem é ascendente ou descendente;
* Abaixo da tabela, exibe-se um texto “Mostrando [#primeiro] a [#último] de [#total] entradas”;
* O usuário pode acessar a ficha detalhada do paciente clicando na linha desejada;
* As entradas inválidas (de paciente nulo ou inválido) serão exibidas em cor vermelha. Se o usuário tentar acessá-las, uma mensagem de erro (alerta) é exibida. O algoritmo utiliza chave primária para acessar a ficha detalhada do paciente;
* A página armazena os possíveis valores de cada atributo. O usuário pode filtrar a busca usando estes valores. Cada atributo ocupa uma seção, e cada valor possui um checkbox. No fim do card, um botão “Filtrar” inicia a busca;
* O usuário pode incluir ou remover filtros usando o link “Mais filtros”.

**Preferências**

* O usuário pode renomear os rótulos desejados. Uma tabela mostra o nome do atributo e um input com o rótulo que já está no arquivo. A página não permite que o rótulo esteja em branco;

**Ficha detalhada**

* A página verifica se há outro paciente com o mesmo número e, caso haja, mostra um alerta antes da tabela, com os links para cada instância repetida;
* Uma tabela mostra todos os pares atributo-valor que estão no arquivo do paciente;
* Um link direciona o usuário diretamente para renomear os atributos;

## Próxima versão

**Painel do usuário (página inicial)**

* Incluir, na primeira seção, link para alterar os dados pessoais do próprio usuário e link para alterar a própria senha;
* Incluir uma terceira seção, apenas para os administradores, com links para cadastrar novo usuário, buscar usuário e redefinir a senha de um usuário;

**Alterar dados pessoais**

* Formulário que altera os dados do próprio usuário. Contém os campos definidos previamente pelo administrador. Um botão valida os dados e envia;

**Alterar senha**

* A página contém três campos de senha: atual, nova senha e repetir nova senha;
* Um botão faz o envio do formulário, que verifica se a senha atual está correta e se os outros dois campos são iguais. Se forem altera a senha do usuário no banco de dados;

**Cadastrar novo usuário**

* Contém um formulário semelhante ao da página “Alterar dados pessoais”, e uma opção autoridade (0-usuário; 1-administrador);
* Ao enviar o formulário, verifica se os campos foram preenchidos. Se sim, uma mensagem é enviada ao e-mail do usuário contendo um link para que ele defina sua senha;