

FACULDADE PROFESSOR MIGUEL ÂNGELO DA SILVA SANTOS – FeMASS

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Título do Trabalho de Conclusão de Curso II

POR:

GUSTAVO ALBERTO DE SOUZA LEMOS

MACAÉ

2022

FACULDADE PROFESSOR MIGUEL ÂNGELO DA SILVA SANTOS – FeMASS

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Gustavo Alberto de Souza Lemos

TÍTULO DO TRABALHO DE CONCLUSÃO DE CURSO II

Trabalho Final apresentado ao curso de graduação em Sistemas de Informação, da Faculdade Professor Miguel Ângelo da Silva Santos (FeMASS), para obtenção do grau de BACHAREL em Sistemas de Informação.

Professor Orientador: Professor Dr. Alan Carvalho Galante

MACAÉ/RJ

2022

GUSTAVO ALBERTO DE SOUZA LEMOS

TÍTULO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Nome do Curso, da Faculdade Professor Miguel Ângelo da Silva Santos (FeMASS), para obtenção do grau de BACHAREL em Nome do Curso.

Aprovada em \_\_\_ de \_\_\_\_\_\_\_ de 20\_\_\_

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. (orientador)

Faculdade Professor Miguel Ângelo da Silva Santos (FeMASS)

1º Examinador

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. (convidado)

Faculdade Professor Miguel Ângelo da Silva Santos (FeMASS)

2º Examinador

CÓPIA DA ATA ASSINADA APÓS A BANCA

**DEDICATÓRIA**

Elemento opcional, colocado após a folha de aprovação. O texto deve ser justificado e espaçamento simples.

**AGRADECIMENTO**

Folha onde o autor faz agradecimentos dirigidos àqueles que contribuíram de maneira relevante à elaboração do trabalho. Elemento opcional, colocado após a dedicatória. O texto deve ser justificado e espaçamento simples.

**EPÍGRAFE**

Folha onde o autor apresenta uma citação, seguida de indicação de autoria, relacionada com a matéria tratada no corpo do trabalho. Usar espaçamento simples e texto justificado.

**RESUMO**

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas e não de uma simples enumeração de tópicos, não ultrapassando 500 palavras, seguido, logo abaixo, das palavras representativas do conteúdo do trabalho, isto é, palavras-chave e/ou descritores. Deve ser apresentado em parágrafo único, com espaçamento simples, resumindo o trabalho.

Palavras-chave: Inserir até 4 palavras. Cada palavra deverá iniciar em maiúsculo e separarem-se por ponto.

**ABSTRACT**

Traduza seu resumo para o inglês

Key words: Inserir até 4 palavras. Cada palavra deverá iniciar em maiúsculo e separarem-se por ponto.

**LISTA DE FIGURAS**

Elemento opcional, que deve ser elaborado de acordo com a ordem apresentada no texto, com cada item designado por seu nome específico, acompanhado do respectivo número da página. Quando necessário, recomenda-se a elaboração de lista própria para cada tipo de ilustração (desenhos, esquemas, fluxogramas, fotografias, gráficos, mapas, organogramas, plantas, quadros, retratos e outros).

Se houver mais de dez figuras é necessário ter lista.

**LISTA DE TABELAS**

Elemento opcional, elaborado de acordo com a ordem apresentada no texto, com cada item designado por seu nome específico, acompanhado do respectivo número da página.

Se houver mais de dez tabelas é necessário ter lista.

**LISTA DE ABREVIATURAS E SIGLAS**

Elemento opcional que consiste na relação alfabética das abreviaturas e siglas utilizadas no texto, seguidas das palavras ou expressões correspondentes grafadas por extenso. Recomenda-se a elaboração de lista própria para cada tipo.

Só é necessário se houver mais de 10 abreviaturas e/ou siglas.

**SUMÁRIO**

De acordo com a NBR 6027, sumário é o último elemento pré-textual, definido como a enumeração das divisões, seções e outras partem de uma publicação, na mesma ordem e grafia em que a matéria nele se sucede. Nele não devem constar os elementos pré-textuais.

**Observação**: Usar o número da primeira página para apresentar a paginação.

[INTRODUÇÃO 12](#_Toc96440250)

[2 TÍTULO DO CAPÍTULO 13](#_Toc96440251)

[3 TÍTULO DO CAPÍTULO 14](#_Toc96440252)

[3.1 Seção 1 14](#_Toc96440253)

[3.2 Seção 2 15](#_Toc96440254)

[4 TÍTULO DO CAPÍTULO 17](#_Toc96440255)

[CONSIDERAÇÕES FINAIS 18](#_Toc96440256)

[REFERÊNCIAS 19](#_Toc96440257)

[ANEXO A 20](#_Toc96440258)

[APÊNDICE A 21](#_Toc96440259)

# 1 INTRODUÇÃO

A emenda constitucional nº19 do artigo 37 parágrafo 3 incentiva a promoção de meios de participação do usuário cidadão, especialmente referentes ao acesso a registros administrativos e às informações sobre atos de governo bem como à prestação de serviços dos órgãos públicos em forma de solicitação e fiscalização.

De acordo com Sant’ana(2009, p.19) é viável a adoção de tecnologias da informação e comunicação (TIC) na participação ativa do usuário cidadão referente à área de administração pública pois, especificamente em prestação de serviços público, existe uma forte demanda tanto nas formas de solicitar atendimento, por parte do cidadão, quanto nas formas de receber e responder um pedido de atendimento, por parte dos servidores públicos.

Sabendo que a Constituição de 1988, por intermédio do artigo 18, garante certo grau de autonomia tanto aos estados quanto aos municípios, algumas cidades já se mobilizaram em relação às formas de interatividade entre cidadão e governo, como exemplo, a cidade de Volta Redonda implantou o aplicativo “Fiscaliza VR” (PREFEITURA DE VOLTA REDONDA, 2019) tornando seu atendimento à população mais acessível e dinâmico.

Macaé também aderiu ao movimento de inovação na interação com a sua população conforme o Decreto Municipal nº 97, de 16 de julho de 2019, inaugurando o laboratório de inovação, sob a responsabilidade da secretaria de planejamento, batizado de Inova Macaé cujo vasto escopo de atividades desempenhadas pela equipe incluem as práticas de melhorias no atendimento ao usuário.

Em menos de um ano dentre várias outras atividades desempenhadas como treinamentos e palestras, os integrantes do Inova também desenvolveram um aplicativo para dispositivos móveis chamado de Macaé App (INOVA MACAÉ,2020) que pode ser obtido através das lojas de aplicativos móveis como “Google Play” ou “Apple Store”.

Diante desse contexto, o Macaé App viabiliza uma aproximação entre o município e os cidadãos de forma que a prefeitura tenha uma participação mais ativa no cotidiano da população tornando-se mais eficiente em observar os problemas despercebidos pela burocratização de processos presentes desde gerações passadas.

O aplicativo serve como uma interface da prefeitura para o cidadão e suas funcionalidades são agrupadas em segmentos como mobilidade urbana, educação, participação cidadã, saúde, Procon, zeladoria urbana, entre outros, para cada segmento é designado um sistema de software devido à alta complexidade do escopo que tal segmento pode apresentar.

Por sua vez, o sistema de software referente à zeladoria urbana, conhecido como “Macaé Fiscaliza”, apresenta uma considerável relevância, pois garante uma comunicação rápida, eficiente e econômica. Através do “Macaé Fiscaliza” os usuários são capazes de notificar o município sobre irregularidades, problemas ou solicitação de serviços nas áreas de infra-estrutura, segurança, saúde, saneamento, meio ambiente, urbanismo, energia, transporte público, limpa e conservação, mobilidade urbana, dentre outros.

Apesar de seu funcionamento ser regular, o sistema interno do “Macaé Fiscaliza” apresenta fragilidades, pois foi desenvolvido por ferramentas que não requerem programação o que o caracteriza não somente “engessado” pela impossibilidade de adaptação às necessidades dos usuários como dependente de ferramentas de terceiros, outra preocupação são as queixas dos funcionários da prefeitura que alegam se submeter ao retrabalho durante sua utilização.

Nesse contexto, como objetivo geral, o presente estudo se compromete em desenvolver um sistema substituto ao existente que a atenda as necessidades dos stakeholders ( funcionários, cidadão ) utilizando conceitos de engenharia de software, arquitetura web, segurança e ferramentas modernas.

Para garantir a efetividade deste compromisso, é necessário: Projetar o sistema que deve atender todo o processo de atendimento do “Macaé Fiscaliza” através das técnicas de análise de requisitos, diagramas UML e “User Stories” para caracterizar as atividades exercidas pelos funcionários; Aplicar a metodologia ágil SCRUM no processo de desenvolvimento de software; Implementar uma aplicação backend que processe a lógica do processo definido pelo projeto do sistema; Aplicar a arquitetura de webservice na aplicação backend tornando possível a comunicação com diversos dispositivos utilizando o ecossistema Spring; Aplicar um framework que lide com as etapas de autenticação e autorização utilizando o protocolo oauth2.1 através do Spring Security; Implementar uma aplicação que sirva de interface para o cidadão permitindo ser acessada através de qualquer dispositivo, aplicando Ionic; Implementar uma aplicação que serva de interface para o servidor público, utilizando angular2+.

A investigação abordada teve o cunho de propor uma solução para os problemas inicialmente apresentados e demonstrá-la em prática, ao classificar esta pesquisa obedecendo a taxonomia estabelecida por PRODANOV(2013, p.128) tratou-se de uma “pesquisa-ação”; mediante ao contexto apresentado, é requerida uma abordagem qualitativa e quantitativa. Para a coleta de dados, a pesquisa se submeteu a sucessivas entrevistas informais com a equipe Inova junto aos técnicos da prefeitura.

Relativo à estruturação do projeto este capítulo serviu como uma apresentação introdutória das justificativas, do problema, objetivos geral e específicos, além da metodologia utilizada.

Em seguida, no segundo capítulo é discutido o referencial teórico sobre: as ferramentas de engenharia de software e gerência de projetos adotadas no projeto do software em questão; arquiteturas web; contexto sobre metodologias ágeis e o SCRUM; processo de autenticação e autorização; tecnologias utilizadas no desenvolvimento (frameworks, linguagens e banco de dados).

No terceiro capítulo é apresentado o contexto do problema da pesquisa abordando o funcionamento do aplicativo de zeladoria urbana “Macaé Fiscaliza” junto das informações obtidas através dos colaboradores entrevistados.

No quarto capítulo, aborda o as etapas do processo de desenvolvimento do sistema e os possíveis obstáculos encontrados.

Por sua vez, o quinto capítulo trata dos testes e validações da aplicação em funcionamento. Por fim, o sexto capítulo apresenta a conclusão e as propostas de trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

**2.1 MÉTODOS ÁGEIS**

**2.2 SCRUM**

**2.3 UML**

**2.4 HTTP**

HyperText Transfer Protoco, HTTP é um protocolo de comunicação que atua na camada de aplicação do padrão de rede de computadores conhecido como o Transmission Control Protocol/Internet Protocol, TCP/IP. O HTTP é descrito como base para a rede mundial de computadores e denominado como “linguagem nativa” dos servidores web. (TANENBAUM, 2011)

Tal comunicação é baseada no envio de requisições através computador, denominado cliente HTTP, por serviços ou dados armazenados em um servidor HTTP que por sua vez processa a requisição e retorna o conteúdo solicitado vinculado a uma resposta. (GOURLEY;TOTTY, 2002, p.4)

O fato de o protocolo ser aplicado à rede mundial de computadores deixa implícito que um conteúdo fornecido pelo servidor HTTP pode ser acessível por clientes HTTP em qualquer parte do mundo e tecnicamente implica que comunicação cliente-servidor é intermediada por outros computadores conforme descrito pela MDN Web Docs(2022), “Entre a solicitação e a resposta existem várias entidades, designadas coletivamente como proxies, que executam operações diferentes e atuam como *gateways*(intermediários) ou caches”.

Para esclarecimento, o termo “cache”, apresentando no parágrafo anterior, pode ser designado para identificar métodos de armazenamento temporário de conteúdos com alta freqüência de acessos, então a utilização de “cache” visa reduzir o tempo de acesso.

**2.4.1 URL**

Qualquer conteúdo ou serviço que pode ser disponibilizado por um servidor específico é um recurso web e para que um recurso possa ser rastreável é adotado um mecanismo conhecido com Unified Resource Locator, URL. (GOURLEY;TOTTY, op. cit.)

O URL é um tipo de URI, Unified Resource Identifier, estruturado pelo esquema de protocolo de comunicação utilizado seguido do endereço do servidor e do caminho (PATH) para acessar o recurso a partir do servidor.

**2.4.2 MÉTODOS HTTP**

Parte integrante de toda requisição responsável por indicar ao servidor qual ação deve ser executada. De acordo com Richardson & Ruby (2007, p.96), dentre o conjunto de métodos disponíveis GET, POST, PUT e DELETE são utilizados nas operações mais comuns.

O método GET é usado em situações que o cliente solicita uma busca por dados, por sua vez POST é utilizado para a postagem, criação de um novo conteúdo no servidor, o PUT serve para a alteração de dados já existentes no servidor enquanto o DELETE excluí algum recurso armazenado no servidor. TANENBAUM (2011, p.686) afirma que os métodos devem ser declarados em letras maiúsculas.

**2.4.3 STATUS HTTP**

Transmitido pelo servidor é a forma de identificar se a requisição foi devidamente processada ou sobre a ocorrência de falhas durante o recebimento, processamento. Conforme a MDN Web Docs (2022), os status são representados em código numérico de três dígitos agrupados em cinco classes: indicadores de informação (100 a 199); indicadores de sucesso(200 a 299); indicadores de redirecionamento(300-399); indicadores de falha no cliente(400 a 499) e indicadores de erro no servidor(500 a 599).

**2.4.4 ESTRUTURA DA MENSAGEM HTTP**

Goruley & Totty, et al(2002, p.10) descrevem que tanto uma requisição HTTP quanto uma resposta HTTP podem ser generalizadas com uma mensagem e que as mensagens trocadas entre cliente e servidor apresentam a mesma estrutura dividida em três partes: linha inicial ou start line; campos de cabeçalho ou header fields; e corpo ou body.

**2.4.4.1 START LINE**

Parte integrante obrigatória cujo propósito varia de acordo com o tipo de mensagem, para uma requisição esta parte deve descrever o método HTTP e o caminho do recurso relativo ao servidor requisitado, enquanto para a resposta esta parte deve ser dedicada a informar o status HTTP.

**2.4.4.2 HEADER FIELDS**

Parte opcional da mensagem a qual geralmente é utilizada para auxiliar no detalhamento da mensagem, cada item do cabeçalho é composto por nome e valor mais um detalhe é que cada item deve ser separado por ponto e vírgula(;).

Os cabeçalhos podem conter dados sobre o cliente, o servidor, o conteúdo transmitido, as credenciais do usuário e muitas outras opções. Richardson & Ruby (2007, p.238) comparam a estrutura de uma mensagem HTTP com um envelope de correspondências assim como os cabeçalhos se comportam como carimbos que acrescentam informações, no caso das correspondências sobre seu trajeto.

**2.4.4.3 BODY**

Parte responsável por armazenar o recurso solicitado, sua obrigatoriedade é situacional.

**2.5 ARQUITETURA**

Conforme o dicionário Michaelis, arquitetura pode ser definida como: “Estrutura, disposição e organização de um conjunto geralmente harmônico”, no contexto deste trabalho o conjunto harmônico é o sistema web proposto, logo delinear a arquitetura de um sistema de software é fundamental para um projeto.

**2.5.1 PADRÕES DE ARQUITETURA**

O desenvolvimento de um sistema de software é suscetível a problemas e Ingeno(2018, p.214) descreve o padrão de arquitetura de software como solução para uma “família” de problemas recorrentes pertencentes a um contexto específico.

Ao sintetizar a definição de Raj, et al(2017, p.13), um padrão de arquitetura serve como molde para a organização estrutural de sistemas complexos de forma a solucionar problemas recorrentes das circunstâncias pré-definidas que o software deve se submeter.

Ademais, através de Fielding (2000, p.6) entende-se que a arquitetura de software remete ao comportamento do software durante sua execução sob uma macro perspectiva ao invés de ser uma estrutura que se resume no código de programação que gera o sistema.

Ingeno (op. cit., p.215) explica que os padrões de arquitetura podem ser aplicados no sistema como um todo ou parcialmente e que mais de um padrão pode ser aplicado no mesmo software, pois um problema pode ser resolvido com uma combinação de soluções.

Quanto a aplicação em um projeto, Raj, et. al(op. cit, p.36) defende o uso de múltiplas arquiteturas afirmando que é uma atitude recomendável por especialistas e personalidades do ramo para acelerar e facilitar o desenvolvimento caso necessário.

E Fielding(op. cit., p.5) esclarece que um sistema pode ser composto em níveis hierárquicos de abstração onde cada nível pode apresentar uma arquitetura de software desde que respeite o nível superior.

Quanto ao conceito de abstração, nas palavras de Shvets (2021, p.15): “Abstração é um modelo de um fenômeno ou objeto do mundo real que representa todos os detalhes relevantes ao contexto, ou perspectiva, e omite os demais detalhes”

**2.5.2 ARQUITETURA MULTICAMADAS**

Antes de prosseguir é fundamental definir o significado de separação de conceitos que, baseado em Ingeno (op. cit., p.183), é um princípio que um sistema atinge quando ele se estrutura em componentes independentes para gerenciar suas funcionalidades de forma a reduzir sua complexidade.

A arquitetura multicamadas decompõe o sistema ou aplicação em elementos menores onde cada um é responsável por parte do conjunto de funcionalidades do sistema que remete a separação de conceitos e torna o sistema adaptável a mudanças. Devido a questões de tradução, uma arquitetura multicamadas pode ser multi-layer ou multi-tier, o que as distingue são suas abordagens e o elemento decomposto.

Uma arquitetura multi-layer separa uma aplicação de software em componentes lógicos, por sua vez uma arquitetura multi-tier segrega um sistema em aplicações (ou programas) distintas as quais podem se localizar em mais de uma máquina física (INGENO, op. cit., p.217). Em outras palavras, enquanto uma arquitetura multi-layer segrega logicamente, a arquitetura multi-tier segrega topologicamente.

É importante esclarecer que um sistema multi-layer organiza seus elementos em camadas hierárquicas onde cada camada fornece serviços para a sua superior. (FIELDING, op. cit., p.46).

**2.5.3 ARQUITETURA CLIENTE-SERVIDOR**

Um sistema que aplica a arquitetura cliente-servidor separa suas funcionalidades técnicas em duas classes de aplicação: frontend e backend (RICHARDS & FORD, 2020, p.121).

O frontend atua como cliente web, servindo como o meio de acesso ao usuário do sistema além de poder processar parte da lógica do negócio, baseado em Ingeno (op. cit., p220), dependendo das limitações do hardware e da necessidade das pessoas envolvidas com o sistema (stakeholders).

E o backend atua como servidor web, geralmente responsável por monitorar mensagens recebidas do frontend e pelo gerenciamento dos dados. Em muitos casos o backend processa a maior parte da lógica da aplicação sob as definições de Ingeno(op. cit., p.220).

Conforme Fielding (op. cit., p.78), a separação de conceitos é à base das regras deste padrão de arquitetura, ou seja, um objetivo fundamental é tornar o sistema tolerável a alterações da mesma forma que a arquitetura multicamadas.

**2.5.4 WEBSERVICES**

No início da internet o acesso aos recursos disponibilizados pelo servidor era viável apenas através de um navegador de internet instalado na máquina do usuário**,** tal restrição impedia o aproveitamento de funcionalidades de servidores já existentes em novas aplicações backend.

Sommerville(2016, p.521) indica que os webservices surgiram da impraticabilidade na comunicação entre servidores caracterizando um webservice como uma aplicação que atende a um padrão de representação de recursos computacionais, ou de dados, destinados ao compartilhamento tanto entre cliente e servidor como também entre servidores distintos.

O mesmo autor (p.524) pontua que a experiências adquiridas com a adesão de webservices culminaram em um conjunto especificações industriais denominadas como “WS-\* standards” que apesar de certificarem a qualidade do software que respeite tais padrões acarretam em alta complexidade além da sobrecarga no uso de recursos computacionais. Essas desvantagens impulsionaram as corporações na adoção de uma abordagem alternativa aos “WS-\* standards”, denominada REST.

**2.5.5 REST**

REST é um acrônimo para Representation State Transfer que nas palavras de Richardson & Ruby(op. cit., p.80) trata-se um conjunto de critérios de projetos a serem aplicados no desenvolvimento de aplicações que se caracterizam como webservice. É uma alternativa muito mais simples que os “WS-\* standards”(SOMMERVILLE,2016,p.530).

Bass, et al.(2021,n.p) declara que o REST consiste de seis restrições impostas na interação entre os componentes do sistema.

* A primeira restrição implica que o sistema deve aplicar a arquitetura cliente-servidor que por conseqüência implica na separação de conceitos discutida no item 2.5.3;
* A restrição seguinte conforme Massé (2012, p.4) exige que a aplicação cliente armazene toda a informação contextual considerada relevante em cada interação com o servidor. Para Fielding( op. cit., p.79) essa informação contextual denomina-se “session state” e por isso o servidor deve ser “stateless”. Designar a complexidade de gerenciamento do estado de sessão para o cliente otimiza a utilização de recursos do servidor tornando-o capaz de atender uma quantidade maior de clientes.
* A terceira restrição se aplica no conteúdo anexado às mensagens de resposta do servidor para que seja identificado como “cacheável” ou “não-cacheável” (Explicação sobre o termo “cache” já explícita no item 2.4). De acordo com Fielding(op. cit., p.80), “cachear” os dados de uma resposta do servidor pode contribuir para uma melhora de performance perceptível ao usuário maior eficiência.
* Massé(op. cit., p.3) explica que a próxima restrição estabelece que haja uniformidade entre as interfaces do servidor, do cliente e das demais entidades intermediárias (proxies e gateways) no processo de troca mensagens. A uniformidade é atingida quando quatro limitações são respeitadas: [1] Os recursos sejam endereçáveis por identificadores únicos como, por exemplo, o URL (item 2.4.1); [2] Um único recurso pode ser representado em diversos formatos de dados através de diversas maneiras para atender as necessidades de qualquer tipo de cliente web; [3] As mensagens devem ser auto-descritivas contendo metadados para dar suporte ao encaminhamento de mensagens realizado pelas entidades intermediárias; [4] À grosso modo, o estado da aplicação deve ser representando como um conjunto de endereços relacionados a ele remetendo ao “Hypermedia as the engine of application state” (HATEOAS).
* A penúltima restrição impõe que o sistema implemente a arquitetura multi-tier (item 2.5.2) em que o servidor, o cliente e as entidades intermediárias atuem em suas respectivas camadas, impossibilitando que cada camada “enxergue” além do seu próprio escopo (FIELDING, op. cit., p.82).
* A última restrição é opcional e permite o cliente baixar recursos executáveis no próprio ambiente da aplicação, como scripts.

Apesar do HTTP apresentar alguns de seus termos utilizados pelas restrições REST e ser o protocolo de comunicação mais adotado por esse conjunto de regras, não há relação de dependência entre REST e HTTP (BASS; et. al., op. cit., p.280).

**2.5.6 REPRESENTAÇÕES DO RECURSO**

Fielding(op. cit., p.90) enuncia que a representação de um recurso é a sequência de bytes (um byte equivale a um agrupamento de oito algarismos binários) junto dos metadados que descrevem esses bytes. A sequência de bytes é o conteúdo que se expressa como documento ou arquivo, enquanto os metadados do conteúdo são o nome do arquivo, tamanho, tipo do arquivo (mime-type ou media-type), data de criação, entre outros.

Recursos que não se adéquam a formatos de arquivo padronizados da computação por se tratarem de dados customizados de acordo com a lógica de negócio de cada sistema costumam a ser formatados em XML ou JSON.

JSON, significa “JavaScript Object Notation” ou notação de objeto javascript, é utilizada na representação e transporte de estruturas de dados genéricas e complexas (Richardson & Ruby, op. cit., p.266). Em relação ao formato XML o JSON é mais compacto, legível além de ser compatível com aplicações cliente escritas na linguagem javascript. Sommerville (2021, p.116) recomenda o uso de JSON como forma de representação de dados em detrimento do XML.

**2.5.7 REST API**

Através da explicação de Massé(op. cit., p.5) entende-se que, no contexto de arquiteturas web, API ou “application programming interface” é um componente integrado ao webservice responsável por interagir com os clientes utilizando um protocolo de comunicação (HTTP, por exemplo) e, de acordo com sua nomenclatura, um REST API é uma API que atende às restrições impostas pelo REST e consiste de um conjunto de elementos interligados.

**2.5.8 ARQUITETURA MVC**

Explicações nos parágrafos a seguir fazem uso do termo “estado” que se traduz como os valores de um conjunto de atributos que caracterizam uma entidade, tal entidade pode ser uma estrutura de dados; um subsistema ou o próprio sistema. Portanto, palavras como “estado da aplicação”, “estado da estrutura de dados”, “estado do componente” fazem referência a este termo.

Sintetizando a descrição de Ingeno (op. cit, p.232), o padrão MVC é aplicável em sistemas frontend e em sistemas backend que apresentam interação direta com o usuário separando a aplicação em três camadas lógicas (multi-layer): Model; View e Controller.

A camada Model engloba um conjunto de componentes que formatam a estrutura dos dados usados na lógica do negócio; armazenam o estado dessa estrutura durante a execução e fornecem funcionalidades para alterar seu estado, essas funcionalidades são acionadas por componentes externos a estrutura em questão.

Já a camada View agrega componentes responsáveis pela interface gráfica, todos os detalhes que o usuário enxerga são descritos pelos componentes desta camada. Qualquer manipulação do usuário com a interface gráfica, como entrada de dados ou cliques de botões devem disparar ações a serem gerenciadas pela camada Controller.

Por sua vez, a camada Controller contém componentes cuja função é basicamente interligar as camadas Model e View, certificando que as ações do usuário se reflitam nos estados dos componentes do tipo Model e que os componentes View reajam de acordo com as interações do usuário.

**2.5.9 ARQUITETURA MVVM**

Variante do modelo MVC, MVVM é aplicável a aplicações frontend e recomendado em casos que exijam interfaces gráficas com reações instantâneas às ações do usuário, como os aplicativos de telefone, aplicações web, etc. Suas camadas lógicas são: Model; View e ViewModel.

A camada Model funciona de forma semelhante a sua contraparte do MVC, porém desta vez não necessita fornecer funcionalidades para alterar seu estado, pois conforme Raj, et. al. (op. cit, p.237) o estado de um componente Model pode ser alterado diretamente através da camada View.

Enquanto a camada View também se assemelha a sua contraparte MVC, entretanto apresenta maior interação com as demais camadas com as camadas Model e ViewModel através do vínculo de dados, ou “data-biding”, responsável pelas atualizações de estado das informações em tempo real. Outro fato é que os elementos desta camada passam a gerenciar parte dos eventos de interface e se comportam dinamicamente, como a validação de dados instantânea. Através da declaração do autor citado no parágrafo anterior, observa-se que apesar do acréscimo de funcionalidades a camada View não se responsabiliza em persistir os dados durante uma atualização de página

A última camada batizada de ViewModel atua de forma semelhante à Controller, além de ser responsável por: gerenciar o estado dos dados; gerenciar a navegação; contém códigos de chamadas ao backend e auxiliar no disparo de eventos da camada View.

**2.6 PADRÕES DE PROJETO**

Com base na definição de Raj, et. al (op. cit., p.13), um padrão de projeto estabelece critérios que delimitam as formas de comportamento e relacionamento interno entre as partes integrantes de um sistema. Cada padrão é aplicável a um contexto específico se propondo a resolver problemas recorrentes tanto durante o desenvolvimento quanto no funcionamento do sistema.

**2.7 SEGURANÇA**

Através da análise de Spilçã (2020, p.4), segurança é um aspecto não-funcional essencial que pode impactar a lucratividade e confiabilidade de um sistema de software caso não levadas em consideração com antecedência.

Diante da vastidão e complexidade do assunto, mecanismos de autenticação e autorização são elementos fundamentais que diferem um sistema minimamente seguro de um sistema totalmente inseguro.

**2.7.1 AUTENTICAÇÃO**

Conforme as afirmações de Sommerville (2021, p.205), trata-se de um processo o qual assegura que o usuário do sistema seja quem ele declara ser. O autor classifica três formas de prover a autenticação em que as informações que cada abordagem exige são fornecidas durante o cadastro.

A primeira é baseada em conhecimento do usuário, onde ao acessar o sistema o usuário deve fornecer as informações como senha ou perguntas pessoais, por exemplo “qual o nome do animal de estimação”, entre outras.

A segunda é baseada em objetos que o usuário possui que podem ser interligados no sistema de autenticação, um exemplo é fornecer o número do telefone em posse do usuário para confirmar sua identidade.

A terceira é baseada nos atributos do usuário, nessa categoria se enquadram métodos que podem se configurar em biometria de digitais, reconhecimento facial, dentre outras.

Ainda através do mesmo autor, apesar da eficácia de cada um dos formatos de autenticação há desvantagens de uso em particular e uma forma de mitigá-los, e até aumentar o nível de segurança, pode ser a exigência de mais de uma abordagem durante a autenticação mesmo assim o nível de autenticação depende das necessidades do software.

**2.7.2 AUTORIZAÇÃO**

Nas palavras de Spilça(op. cit, p.15), autorização é um processo que permite o acesso funcionalidades ou dados específicos através da verificação das permissões de um usuário autenticado. Para Sommerville(2021, p.211), as permissões do usuário são estabelecidas obedecendo a política de controle de acesso submetida à lógica do negócio.

**2.7.3 JWT**

Acrônimo para JSON Web Token,

**2.7.4 PROTOCOLO OPEN AUTHORIZATION**

**2.8 FERRAMENTAS**

# TÍTULO DO CAPÍTULO

Aqui deve constar todo o capítulo desenvolvido com as seções necessárias.

Provavelmente este será capítulo do autor, onde será explanada toda a sua pesquisa. Entretanto, pode ser que você queira estruturar sua pesquisa em mais de um capítulo.

## Seção 1

São consideradas ilustrações: desenhos, gráficos, fluxogramas, fotografias, figuras, mapas, organogramas, enfim imagens que acompanhem um texto. As ilustrações devem ser inseridas o mais próximo possível do trecho a que se refere, conforme projeto gráfico do trabalho.

A identificação das ilustrações deve aparecer na parte superior, precedida da palavra designativa, seguida de seu número de ordem de ocorrência no texto, em algarismos arábicos e do respectivo título, usando a mesma tipologia de fonte utilizada para as seções primárias do trabalho. Após a ilustração, na parte inferior, indicar obrigatoriamente a fonte (ainda que seja produção do próprio autor), utilizando fonte tamanho 10, estilo regular e espaçamento simples.

**QUADRO 1 - CARACTERÍSTICAS DAS CONTABILIDADES FINANCEIRA E GERENCIAL**

|  |  |  |
| --- | --- | --- |
|  | **Contabilidade Financeira** | **Contabilidade Gerencial** |
| **Clientela** | Externa: acionistas, credores, autoridades tributárias | Interna: funcionários, administradores, executivos |
| **Propósito** | Reportar o desempenho passado às partes externas; contratos com proprietários e credores | Informar decisões internas tomadas pelos funcionários e gerentes; feedback e controle sobre desempenho operacional; contratos com proprietários e credores |
| **Data** | Histórica, atrasada | Atual, orientada para o futuro |
| **Natureza da**  **Informação** | Objetiva, auditável, confiável, consistente, precisa | Mais subjetiva e sujeita a juízo de valor, válida, relevante, acurada |
| **Escopo** | Muito agregada; reporta toda a empresa | Desagregada; informa as decisões e ações locais |

Fonte: Adaptado de Atkinson et al. (2000, p. 38).

São considerados QUADROS as apresentações de tipo tabular, ou seja, compostos por linhas e **colunas com uma moldura (um quadrado) em torno**. São empregados para apresentar conteúdo teórico, como classificações, comparações e dados numéricos sem tratamento estatístico. Dados com tratamento estatístico devem ser apresentados em TABELAS.

## Seção 2

Conforme o conteúdo que apresentam, as tabelas podem ser estatísticas de codificação, de conversão de unidades técnicas, de rotina ou controle e especiais.

O título da tabela deve ser inscrito no topo e deve indicar a natureza e as abrangências: geográfica e temporal dos dados numéricos.

**TABELA 1 - OPERADORAS REGISTRADAS SEGUNDO REGIÃO DA SEDE E CLASSIFICAÇÃO**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classificação da Operadora | Norte | Nordeste | Sudeste | Sul | C. Oeste | Total |
| Administradora de Planos | 0 | 1 | 7 | 3 | 1 | 11 |
| Administradora de Serviços | 0 | 2 | 2 | 1 | 0 | 5 |
| Autogestão Não Patrocinada | 3 | 22 | 67 | 23 | 15 | 130 |
| Autogestão Patroc. Multipatrocinada | 1 | 3 | 15 | 10 | 6 | 35 |
| Autogestão Patroc. Multipatrocinada / RH | 0 | 0 | 1 | 1 | 1 | 3 |
| Autogestão Patrocinada Singular | 0 | 7 | 18 | 2 | 10 | 37 |
| Autogestão Patrocinada Singular / RH | 5 | 5 | 95 | 21 | 10 | 136 |
| Cooperativa Médica | 18 | 67 | 183 | 66 | 36 | 370 |
| Cooperativa odontológica | 6 | 24 | 99 | 31 | 12 | 172 |
| Filantropia | 2 | 9 | 97 | 20 | 1 | 129 |
| Medicina de Grupo | 25 | 99 | 483 | 128 | 40 | 775 |
| Odontologia de Grupo | 8 | 72 | 274 | 88 | 18 | 460 |
| Seguradora Especializada em Saúde\* | 0 | 0 | 13 | 1 | 0 | 14 |
| Total | 68 | 311 | 1.354 | 395 | 149 | 2.277 |

Fonte: Agência Nacional de Saúde Complementar (2004)

\*A região noroeste do Brasil não possui seguradoras especializadas em saúde.

A fonte deve ser indicada logo abaixo da tabela. Esta informação é obrigatória ainda que a tabela tenha sido elaborada pelo autor.

Quando uma tabela ocupar mais de uma folha, não será delimitada na parte inferior, repetindo-se o cabeçalho e o título na folha seguinte. Cada folha deve ter as seguintes indicações: continua (na primeira), conclusão (na última) e continuação (nas demais). As indicações de fontes e notas devem aparecer na folha de conclusão da tabela.

Caso algum valor tabulado mereça explicação, pode-se acrescentar nota explicativa com um asterisco logo abaixo da tabela.

A estrutura da tabela é constituída de traços (retas perpendiculares) e é delimitada em sua parte superior e na parte inferior por traços horizontais paralelos. **Não delimitar (fechar) com traços verticais as laterais direita e esquerda**.

3.3 Seção 3

Figura é a denominação genérica atribuída aos gráficos, fotografias, gravuras, mapas, plantas, desenhos ou demais tipos ilustrativos. Quando a figura for representada apenas por gráficos, a denominação pode ser feita por esta palavra (gráfico), nas demais utilize figura.

FIGURA 1 – PIRÂMIDE DE NECESSIDADES DE MASLOW



Fonte: Adaptado de Andrade (2011, p. 123)

# DESENVOLVIMENTO

Seguindo a metodologia SCRUM todos as user stories que foram mencionadas no capítulo anterior em um product backlog estão sub-agrupadas em 5 sprints, deve ser reforçado que cada sprint foi uma etapa do desenvolvimento contendo seus artefatos correspondentes finalizados (user stories, modelo de classes, protótipo de telas, código backend e código front-end) durante o processo, as quais são descritas a seguir:

# Sprint 1

# User Stories

Para o primeiro sprint de desenvolvimento foram selecionadas as seguintes user stories, cada user story foi subdividida em subtarefas menores:

* Como cidadão, gostaria de poder criar conta no sistema sem precisar vincular minha contas de provedores como o google ou redes sociais ao sistema.
  + Criar projeto backend
  + Criar classe model para cidadão
  + Criar Data Access Object para cidadão
  + Criar Controller de cidadão
  + Adicionar método de cadastro para cidadão
  + Adicionar método para cidadão logar
  + Criar tela de cadastro para o cidadão
  + Criar tela de login para o cidadão
* Como cidadão, gostaria de acessar o sistema pelo computador tradicional
  + Criar projeto frontend

# Modelo de Classes



Fonte: Autor, 2022

# Protótipo de Telas

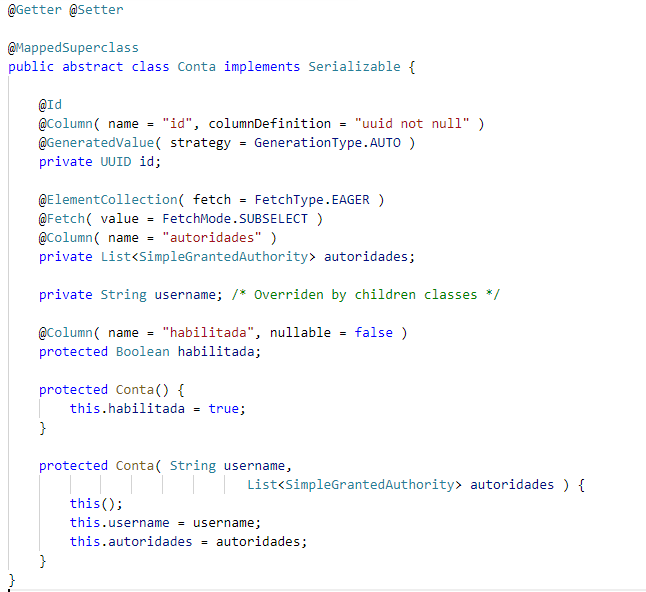
FIGURA 1 – Protótipo da tela de login



Fonte: Autor, 2022

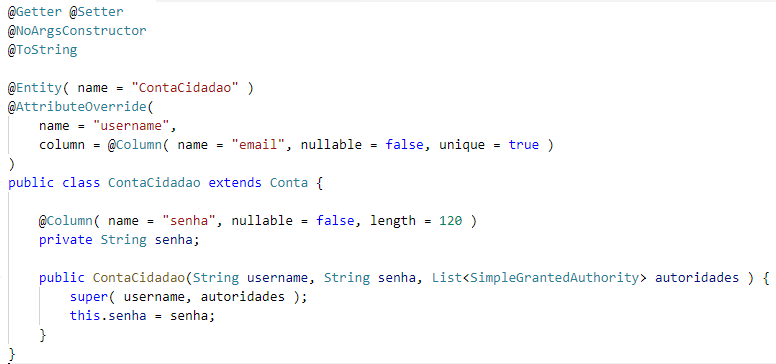
# Código Fonte – Backend

FIGURA 2 – Exemplo da classe Conta



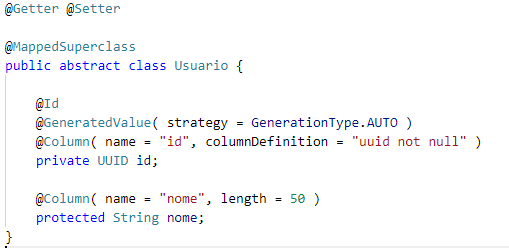
Fonte: Autor, 2022

FIGURA 3 – Exemplo da classe ContaCidadao



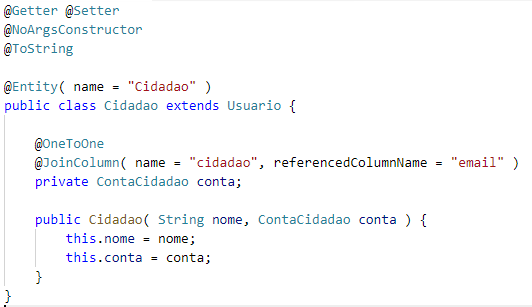
Fonte: Autor, 2022

FIGURA 4 – Exemplo da classe Usuario



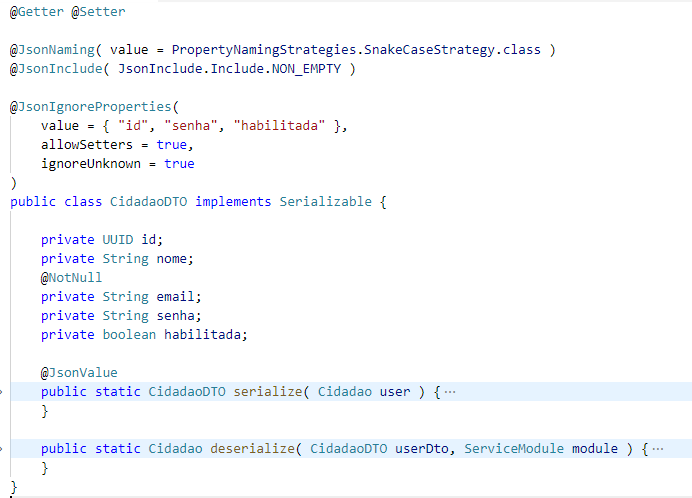
Fonte: Autor, 2022

FIGURA 5 – Exemplo da classe Cidadao



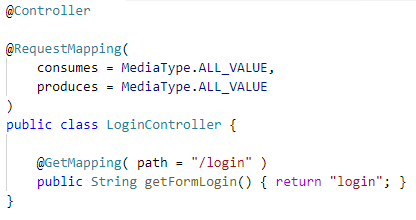
Fonte: Autor, 2022

FIGURA 6 – Exemplo da classe CidadaoDTO



Fonte: Autor, 2022

FIGURA 7 – Exemplo da classe Login Controller



Fonte: Autor, 2022

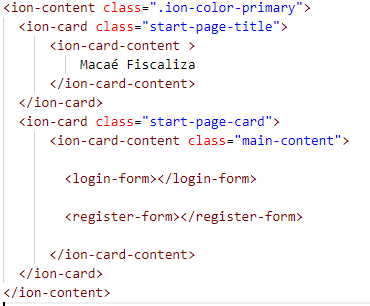
FIGURA 7 – Exemplo do formulário de login



Fonte: Autor, 2022

# Código Fonte - Frontend

FIGURA 8 – Exemplo da pagina inicial do aplicativo



Fonte: Autor, 2022

FIGURA 9 – Exemplo do template form-login



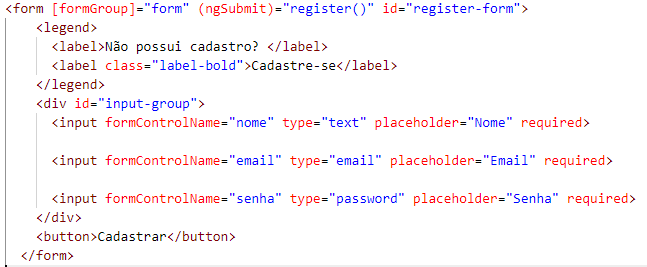
Fonte: Autor, 2022

FIGURA 10 – Exemplo da lógica do componente login-form



Fonte: Autor, 2022

FIGURA 11 – Exemplo do template register-form



Fonte: Autor, 2022

FIGURA 9 – Exemplo da lógica do componente register-form



Fonte: Autor, 2022

# Sprint 2

# Modelo de Classes

# Protótipo de Telas

# Código Fonte – Backend

# Código Fonte - Frontend

# Sprint 3

# Modelo de Classes

# Protótipo de Telas

# Código Fonte – Backend

# Código Fonte - Frontend

# Sprint 4

# Modelo de Classes

# Protótipo de Telas

# Código Fonte – Backend

# Código Fonte - Frontend

# Sprint 5

# Modelo de Classes

# Protótipo de Telas

# Código Fonte – Backend

# Código Fonte - Frontend

1. **TESTES**

Aqui deve constar todo o capítulo desenvolvido com as seções necessárias. Caso seja desnecessário este capítulo, basta excluí-lo. No caso de mais capítulos, basta inseri-los e seguir a numeração.

# CONSIDERAÇÕES FINAIS

Aqui devem constar as considerações finais do trabalho, podendo retomar o objetivo proposto, se este foi alcançado e trabalhos futuros a partir do que foi desenvolvida na pesquisa que resultou no TCC.

# REFERÊNCIAS

Listar, de acordo com as normas da ABNT, os materiais que foram utilizados naelaboração do Trabalho de Conclusão de Curso.

**Observação:** Como qualquer trabalho acadêmico, o TCC precisa ser fundamentado. Portanto, faça citações (diretas ou indiretas) no corpo do texto, de acordo com as normas da ABNT.

Pode-se utilizar o site <http://www.rexlab.ufsc.br:8080/more/> que gera as referências no padrão da norma ABNT e também as citações.

Dicas de como realizar citações podem ser vistas no site: <http://www.tecmundo.com.br/tutorial/834-aprenda-a-usar-as-normas-da-abnt-citacao-2-de-4-.htm>).

ALVES-MAZZOTTI, A. J.; GEWANDSZNAJDER, F. **O método nas ciências naturais e sociais**: pesquisa quantitativa e qualitativa. São Paulo: Pioneira, 1998.

BIBLIOTECA NACIONAL (Brasil). **Bibliografia do folclore brasileiro**. Rio de Janeiro: Divisão de Publicações, 1971.

LAWRENCE, P. R. **Changingoforganizationalbehaviorpatterns**. Piscataway:

TransactionPublishers, 2001. Disponível em:

<http://search.epnet.com/direct.asp?an=7511860&db=buh>. Acesso em: 22 maio

2004.

MARTONE, C. L. Modelo básico para economia fechada e aberta. In: LOPES, L. M.;

VASCONCELLOS, M. A. S. (Org.). **Manual de macroeconomia**: nível básico e nível

intermediário. 2. ed. São Paulo: Atlas, 2000. cap. 13, p. 299-311.

SIQUEIRA, N. A. S. **Substituição da força de venda própria por distribuidores**:

um estudo de caso. 2002. 125 f. Dissertação (Mestrado em Administração de

Empresas)–Fundação Escola de Comércio Álvares Penteado - FECAP, São Paulo,

2002.

WOOD JUNIOR, T. **Executivos neuróticos, empresas nervosas**. São Paulo: Negócio, 2002.

# ANEXO A

Elemento opcional. O(s) anexo(s) é (são) identificado (s) por letras maiúsculas consecutivas, travessão e pelo (s) respectivo (s) título (s). Excepcionalmente utilizam-se letras maiúsculas dobradas, na identificação dos anexos, quando esgotadas as 23 letras do alfabeto.

Cada anexo deverá ser colocado em uma página.

Anexos são documentos ou materiais adquiridos em órgãos do governo, empresas ou outras fontes, que não foram construídos pelo autor deste trabalho, mas foram utilizados na pesquisa.

Cabe um texto introdutório direcionando para o quê será encontrado neste capítulo, seguido dos anexos, cada um em uma nova página, numerados segundo a ABNT.

# APÊNDICE A

Elemento opcional. O (s) apêndice (s) é (são) identificado (s) por letras maiúsculas consecutivas, travessão e pelo (s) respectivo (s) título (s). Excepcionalmente utilizam-se letras maiúsculas dobradas, na identificação dos apêndices, quando esgotadas as 23 letras do alfabeto.

Apêndices são documentos ou materiais feitos pelo autor deste trabalho e que ilustram mais a pesquisa. Pode ser um formulário criado para levantamento de campo, o código de um programa, um texto de algum assunto que não seja central do trabalho (senão deve estar no capítulo do autor), alguma explanação periférica que serve como complementariedade.

Por exemplo, os dados tabulados de uma pesquisa de campo devem estar no trabalho bem como as conclusões tiradas. Mas o formulário utilizado para o levantamento dos dados pode estar aqui, para uma visualização dos campos considerados na pesquisa.

Cabe um texto introdutório direcionando para o quê será encontrado neste capítulo, seguido dos apêndices, cada um em uma nova página, numerados segundo a ABNT.

Cada apêndice deve ser colocado em uma página.