INSTITUTO INFNET

Pedro Catão Pereira da Silva

DESENVOLVIMENTO WEB COM DOCKER E ASP.NET CORE   
CÁPSULA DO TEMPO

Rio de Janeiro

2019

Pedro Catão Pereira da Silva

DESENVOLVIMENTO WEB COM DOCKER E ASP.NET CORE   
CÁPSULA DO TEMPO

Trabalho de conclusão de curso apresentado à Escola Superior da Tecnologia da Informação do Instituto Infnet como requisito parcial à obtenção do grau de graduação em Análise e Desenvolvimento de Sistemas.

Orientador: Gustavo de Miranda

Rio de Janeiro

2019

|  |
| --- |
| Ficha catalográfica : Procurar a biblioteca pelo e-mail: biblioteca@infnet.edu.br |

Pedro Catão Pereira da Silva

DESENVOLVIMENTO WEB COM DOCKER E ASP.NET CORE   
CÁPSULA DO TEMPO

Trabalho de conclusão de curso apresentado à Escola Superior da Tecnologia da Informação do Instituto Infnet como requisito parcial à obtenção do grau de graduação em Análise e Desenvolvimento de Sistemas.

Aprovado em:

BANCA EXAMINADORA

Nome do Orientador - Orientador

Instituto Infnet

Nome do professor avaliador 1

Instituto Infnet

Nome do professor avaliador 2

Instituto Infnet

# AGRADECIMENTOS

Agradeço aos meus pais Tê e Catão pelo apoio e fé de que um dia o seu filho se formasse. À minha mulher e companheira Paula, ao meu Orientador Gustavo de Miranda que acreditou no meu projeto, à Professora Talita Peixoto que me fez acreditar que era possível. Ao Instituto Infnet e ao professor Erico Torres, que me fizeram gostar de desenvolvimento de sistemas. E a todos os outros professores que pavimentaram o meu caminho até aqui.

# RESUMO

O presente trabalho tem como objetivo criar um aplicativo Web de Cápsula do Tempo usando as tecnologias mais recentes do mercado e pensado para ser um aplicativo rodando na nuvem, altamente escalável, altamente disponível e multiplataforma.

Esse aplicativo tem como objetivo permitir que um usuário crie uma Cápsula do Tempo contendo uma imagem, uma mensagem e uma data de abertura. Ao alcançar a data de abertura a cápsula ficará aberta para consulta o mesmo período que esta ficou fechada, permitindo que depois de algum tempo aquela cápsula seja reciclada, e usada por outro usuário.

Outro objetivo é demonstrar as tecnologias utilizadas e suas interações, bem como estabelecer uma forma de referência no desenvolvimento de aplicativos Web para a nuvem no ano de 2019.

Palavras-chave: Cápsula do Tempo, ASP.NET Core, Docker.

# ABSTRACT

The purpose of this paper is to create a Time Capsule Web application using the latest and most accurate technologies for a highly scalable, highly available and cross-platform cloud-running application.

This application is intended to allow a user to create a time capsule with an image, a message, and an opening date. When accessing the opening data of an open session for consultation in the same period, it remained complete, allowing for some time the capsule to be recycled, and used by another user.

Other objective of this paper is to show the technologies used and their interactions, as well as a reference form in the development of Web applications for the cloud in the year 2019.

Keywords: Time Capsule, ASP.NET Core, Docker.

# LISTA DE ABREVIATURAS E SÍMBOLOS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| HTTP |  |  |  |  |  | HyperText Transfer Protocol |
| IDE |  |  |  |  |  | Integrated Development Environment ou Ambiente de Desenvolvimento Integrado |
| API |  |  |  |  |  | Application programming interface |
| MVC |  |  |  |  |  | Model-view-controller |
| OS |  |  |  |  |  | Operating system ou Sistema operacional |
| URI |  |  |  |  |  | Uniform Resource Identifier |
| SQL |  |  |  |  |  | Structured Query Language |
| UI |  |  |  |  |  | User Interface |
| DDD |  |  |  |  |  | Domain-Driven Design |
| Open Source |  |  |  |  |  | Software livre |
| SOLID |  |  |  |  |  | Mnemonic acronym for five design principles intended to make software designs more understandable |
| MSDN |  |  |  |  |  | Microsoft Developer Network |
| CAPTCHA |  |  |  |  |  | Completely Automated Public Turing test to tell Computers and Humans Apart |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# LISTA DE FIGURAS

[Figura 1 - Arquitetura Docker 16](#_Toc532329363)

[Figura 2 - Arquivo Dockerfile 16](#_Toc532329364)

[Figura 3 - Verbos HTTP e suas Utilizações 18](#_Toc532329365)

[Figura 4 - Estratégia de Branchs GitFlow 20](#_Toc532329366)

[Figura 5 - Interface DONTPAD 22](#_Toc532329367)

[Figura 6 - Interface PASTEBIN 24](#_Toc532329368)

[Figura 7 - Interface do repositório de Cápsula do Tempo no Domínio 28](#_Toc532329369)

[Figura 8 - Implementação do Repositório de Cápsula do Tempo 29](#_Toc532329370)

[Figura 9 - Estrutura do código 30](#_Toc532329371)

[Figura 10 - Casos de Uso 31](#_Toc532329372)

[Figura 11 - Classe CapsulaDoTempoService 31](#_Toc532329373)

# SUMÁRIO

[AGRADECIMENTOS 5](#_Toc989403)

[RESUMO 6](#_Toc989404)

[ABSTRACT 7](#_Toc989405)

[LISTA DE ABREVIATURAS E SÍMBOLOS 8](#_Toc989406)

[LISTA DE FIGURAS 9](#_Toc989407)

[SUMÁRIO 10](#_Toc989408)

[1. INTRODUÇÃO 12](#_Toc989409)

[1.1. TEMA 12](#_Toc989410)

[1.2. OBJETIVO 13](#_Toc989411)

[1.3. ESTRUTURA DO DOCUMENTO 14](#_Toc989412)

[2. EMBASAMENTO TEÓRICO 15](#_Toc989413)

[2.1. TECNOLOGIA UTILIZADA 15](#_Toc989414)

[2.1.1. DOCKER 16](#_Toc989415)

[2.1.2. ASP.NET CORE 18](#_Toc989416)

[2.1.3. VISUAL STUDIO 18](#_Toc989417)

[2.1.4. WEB API 18](#_Toc989418)

[2.1.5. GIT 20](#_Toc989419)

[2.1.6. GIT HUB 21](#_Toc989420)

[2.1.7. DOCKER HUB 21](#_Toc989421)

[2.1.8. MONGO DB 22](#_Toc989422)

[3. TRABALHOS RELACIONADOS 23](#_Toc989423)

[3.1 DONT PAD 23](#_Toc989424)

[3.2 SNAPCHAT 24](#_Toc989425)

[3.3 PASTEBIN 25](#_Toc989426)

[4. SISTEMA 26](#_Toc989427)

[4.1. VISÃO GERAL 26](#_Toc989428)

[4.1.1. API 27](#_Toc989429)

[4.1.2. WEB UI 27](#_Toc989430)

[4.2. ARQUITETURA 27](#_Toc989431)

[4.2.1. CAMADA DE INFRAESTRUTURA - PERSISTÊNCIA OU CAMADA DE DADOS 29](#_Toc989432)

[4.2.2. CAMADA DE DOMÍNIO 32](#_Toc989433)

[4.2.3. CASOS DE USO 33](#_Toc989434)

[5. CONCLUSÃO 34](#_Toc989435)

[REFERÊNCIAS BIBLIOGRÁFICAS 36](#_Toc989436)

[SUGESTÃO DE TRABALHOS FUTUROS 37](#_Toc989437)

[ANEXO I – CASOS DE USO 38](#_Toc989438)

[ANEXO II – DIAGRAMAS DE SEQUÊNCIA 40](#_Toc989439)

# 1. INTRODUÇÃO

## 1.1. TEMA

Este trabalho visa demonstrar o desenvolvimento de um aplicativo WEB com alta escalabilidade usando ASP.NET Core 2.1 e Docker. O Tema de Cápsula do Tempo foi escolhido para ser desenvolvido usando estas tecnologias.

Utilizando as tecnologias descritas acima, podemos criar um aplicativo Web para ser hospedado na nuvem, inclusive simultaneamente em várias nuvens diferentes.

## 1.2. OBJETIVO

O presente trabalho tem como objetivo desenvolver um aplicativo Web altamente escalável e altamente disponível na nuvem. O aplicativo funcionará como uma Cápsula do Tempo, na qual o usuário irá acessar um endereço qualquer no subdomínio do aplicativo em questão e criar uma Cápsula do Tempo, com uma imagem, uma mensagem e uma data de abertura. Qualquer usuário pode consultar qualquer Cápsula do Tempo, mas elas não estarão indexadas em nenhum lugar. Caso a data de abertura da cápsula seja uma data futura, o usuário não poderá ver o conteúdo da cápsula e, caso a cápsula esteja aberta, o conteúdo será apresentado para o usuário.

Para o desenvolvimento do aplicativo em questão, serão utilizadas as tecnologias mais recentes no momento. O ASP.NET Core será utilizado como framework de desenvolvimento web, o Docker será utilizado como ambiente para a publicação do aplicativo e o MongoDB será utilizado como base de dados. Todo o código é aberto e de livre utilização e será disponibilizado no GitHub.

Como conclusão, é esperado que exista uma aplicação Web com as características descritas em funcionamento na nuvem, com o código disponibilizado no GitHub, e a imagem Docker disponibilizada no DocketHub.

## 1.3. ESTRUTURA DO DOCUMENTO

Este projeto será estruturado em capítulos que visam amplificar o entendimento do produto proposto. Os capítulos serão descritos a seguir:

O “Capítulo 1 – Introdução” abre a leitura e apresenta a pesquisa em seus aspectos estruturantes, tais como o tema e o objetivo pretendido; o “Capítulo 2 - Embasamento Teórico” relaciona-se ao conjunto de tecnologias que o projeto utiliza, descrevendo-as individualmente; o “Capítulo 3 - Trabalhos Relacionados” apresenta produtos e trabalhos similares ao desenvolvido nessa pesquisa, que amplificam o entendimento da área de conhecimento abordada; o “Capítulo 4 – Sistema” relata e documenta o desenvolvimento do sistema e sua arquitetura, apresentando o produto final; e no “Capítulo 5 – Conclusão” - o texto coloca em diálogo o resgate das principais questões de pesquisa propostas na introdução, sobretudo o problema e o objetivo, e apresenta o produto final desenvolvido, verificando se essas questões foram contempladas.

# 2. EMBASAMENTO TEÓRICO

Nesse capítulo, veremos as tecnologias-base para a construção do sistema que é mostrado neste projeto e motivos que envolvem sua utilização.

## 2.1. TECNOLOGIA UTILIZADA

O aplicativo de cápsula do tempo foi desenvolvido em ASP.NET Core 2.1 utilizando o Visual Studio 2017 como IDE. Para o armazenamento dos dados, foi escolhido o MongoBD, um banco de dados não relacional escalável e amplamente utilizado pela comunidade de desenvolvedores.

As funcionalidades do sistema estão disponibilizadas para uso para diversas camadas de apresentação. Isso foi possível graças ao conceito de WEB API, disponibilizando os recursos do sistema por meio do protocolo HTTP. O ASP.NET WEB API foi escolhido como framework de desenvolvimento e o ASP.NET MVC foi escolhido como a primeira camada de apresentação do sistema.

Pensado desde o início em sua escalabilidade, o projeto é suportado pelo Docker, um conceito parecido com a virtualização, mas com algumas diferenças que serão abordadas mais à frente.

O código foi versionado utilizando o Git e hospedado no GitHub.

As imagens Docker geradas foram hospedadas no DockerHub e estão disponibilizadas abertamente para download.

### 2.1.1. DOCKER

Primeiro, é preciso entender o problema que é solucionado pelo Docker. Para escalar uma aplicação, ou seja, pra ela poder receber mais requisições em um mesmo período de tempo, uma solução simples é aumentar o número de máquinas que estão processando aquelas requisições. Mas montar uma máquina é muito custoso, imagine toda vez que a demanda aumentar seja necessário comprar uma máquina, instalar um OS, fazer o deploy da sua aplicação e de todos os pré-requisitos, ligar a sua máquina na rede para aí sim a aplicação conseguir processar mais requisições.

Para resolver esse problema, o conceito de virtualização foi criado. Nele, uma máquina é hospedada virtualmente em uma máquina física que está rodando e, com isso, pode ser criada uma imagem da máquina a ser levantada, com a rede, OS, aplicação e os pré-requisitos instalados previamente, e essa máquina pode subir em um parque de máquinas físicas sempre que for preciso. Porém, essa solução ainda apresenta alguns problemas que o Docker soluciona: o boot de um OS é demorado e a máquina hospedeira perde parte de seu poder de processamento ao executar operações do OS de várias máquinas virtuais, além do próprio OS da máquina hospedeira.

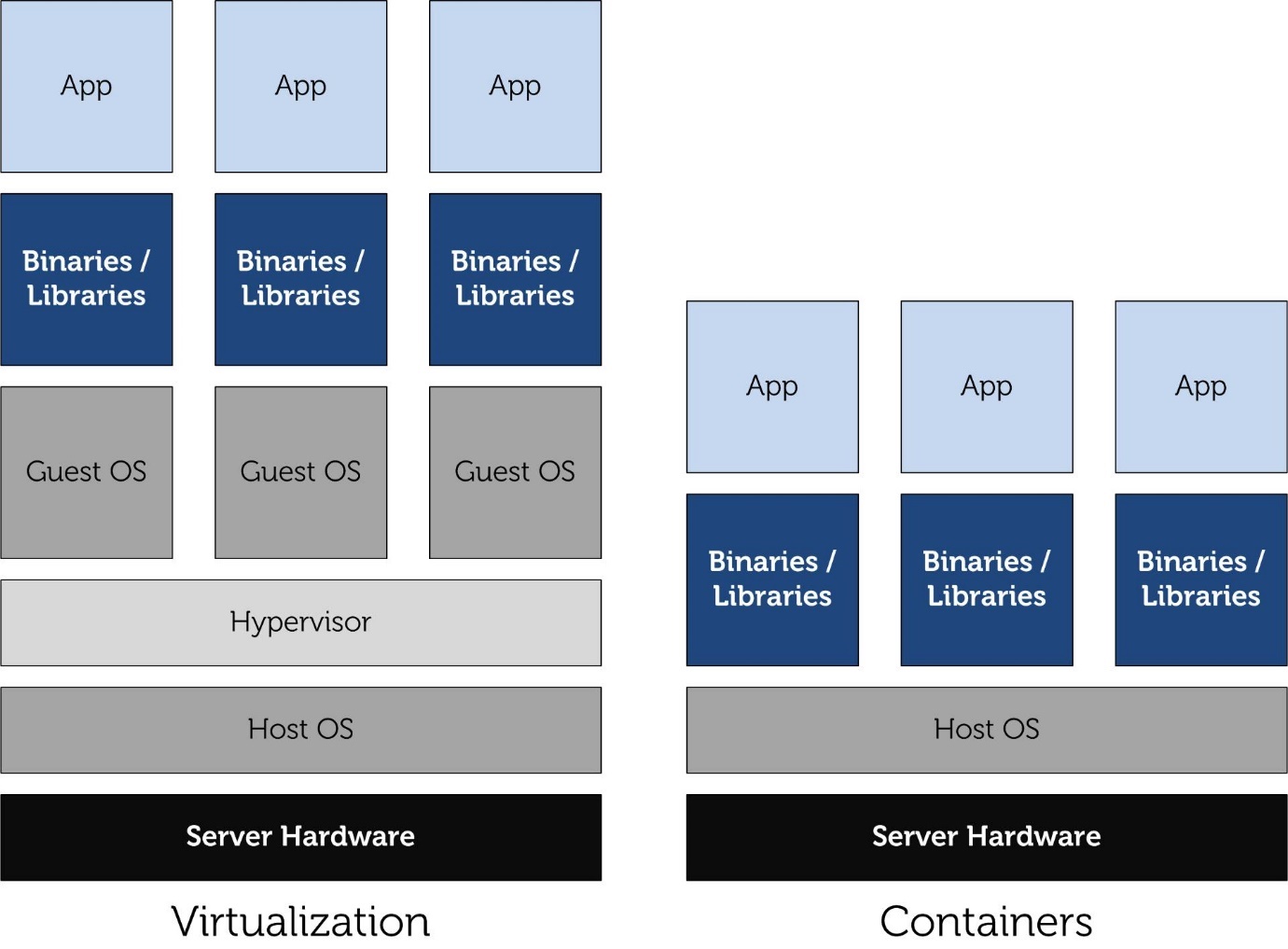


Figura 1 - Arquitetura Docker

<https://www.mundodocker.com.br/o-que-e-container/>

Com o conceito de contêinerização, apenas um OS está rodando na máquina e as aplicações rodam no contêiner, diretamente sobre o OS da maquina física. Portanto, quando uma aplicação precisa de mais poder de processamento, novos contêineres são levantados rapidamente.

As instruções para a criação de um container estão em um arquivo chamado “Dockerfile”.

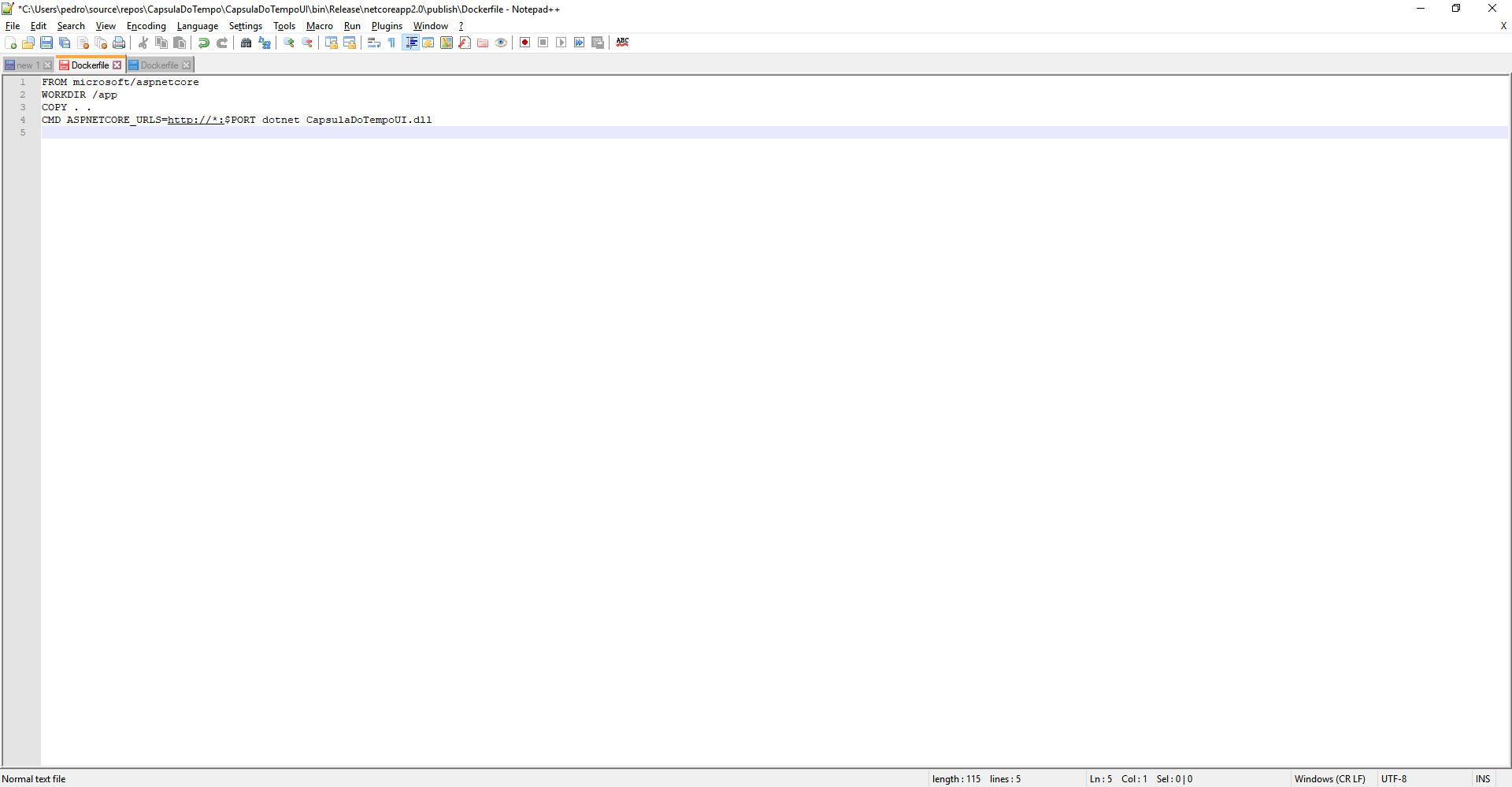


Figura 2 - Arquivo Dockerfile

Neste arquivo de exemplo, foi definida a imagem que será utilizada como base, no qual a pasta de trabalho é a /app. O mesmo foi configurado para copiar todo o conteúdo da imagem e o comando de expor a CapsulaDoTempoUI.dll no protocolo HTTP em qualquer porta.

### 2.1.2. ASP.NET CORE

O ASP.NET Core é um framework de desenvolvimento web, multiplataforma open-source baseado no .NET Core, uma releitura do bem-sucedido .NET Framework, mas pensado e reescrito desde o início para ser um framework multiplataforma, mais leve, mais rápido, mais simples e não dependente de nenhuma API do Windows.

### 2.1.3. VISUAL STUDIO

O Visual Studio é a IDE padrão de desenvolvimento .NET da Microsoft. Possui muitas integrações com diversos serviços que auxiliam o desenvolvimento de software, possui os mais diversos plugins e é amplamente utilizado no desenvolvimento de software, principalmente no .NET Framework.

Integrado a quase todos os serviços Microsoft, o Visual Studio permite ao desenvolvedor publicar seus aplicativos diretamente na nuvem da Microsoft, a Azure. Integrado também ao Docker, o Visual Studio facilita o desenvolvimento e depuração de aplicativos no Docker.

### 2.1.4. WEB API

Este estilo arquitetural possui princípios básicos baseados no protocolo HTTP tais como:

* Nomear seus recurso através de URIs globais, que funcionam como identificadores único destes. Recurso, por sua vez, pode ser definido como qualquer coisa que seja importante o suficiente para necessitar de uma referência;
* Utilização do conjunto de verbos padrões do protocolo HTTP (como os mais comuns GET, POST e os menos utilizados PUT e DELETE) (figura 3), para que, através das chamadas às URIs, comuniquem para o serviço a ação que deverão executar com o recurso. As informações trocadas nessa comunicação entre cliente e serviço são definidas por meio de declarações em seu cabeçalho sobre o tipo de mídia utilizada (XHTML, JPEG, PNG e a mídia que mais será utilizado nesse projeto, JSON);
* Vinculação de serviços por meio de hipermídia. Um recurso pode disponibilizar links de acessos a outros recursos da mesma ou de outra aplicação; e
* Os recursos são autossuficientes e toda informação necessária para processá-los estará contida nela mesma, evitando assim a manutenção do estado, além de afetar a escalabilidade e a quantidade de clientes que interagem com o servidor.

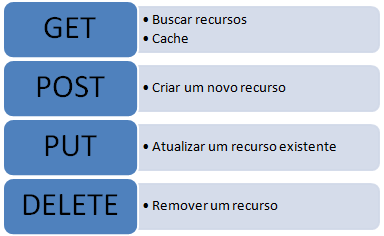


Figura 3 - Verbos HTTP e suas Utilizações

<http://msdn.microsoft.com/pt-br/library/dd941696.aspx>.

### 2.1.5. GIT

O GIT é um versionador de código fonte distribuído. Com ele o desenvolvedor pode rastrear o desenvolvimento do software ao longo do tempo, adotar estratégias de promoção de código e criar forks de um código, entre outros. Criado em 2005, hoje é amplamente utilizado pela comunidade de desenvolvimento de software.

O aplicativo de Cápsula do Tempo utilizou o GIT como versionador de código fonte. Cada alteração de código, desde a criação do aplicativo, pode ser consultada e recuperada. Outros desenvolvedores podem pegar o código atual e criar as suas próprias versões do aplicativo (Forks) e também podem ajudar no desenvolvimento de novas funcionalidades.

O aplicativo de Cápsula do Tempo utilizou a estratégia de branches GitFlow, que consiste em ter basicamente dois branches principais durante todo o ciclo de desenvolvimento. O master é o código referente à versão de produção e o develop é, o código referente à versão de desenvolvimento. Cada novo desenvolvimento deve ser feito em uma feature branch, que é uma branch do develop na qual o desenvolvedor vai criar o seu código. Após o desenvolvimento e teste desta feature, é feito o merge para develop.

Quando os desenvolvedores julgarem que uma nova versão pode ser publicada, então é criada uma branch de release, a partir do develop. Esta branch de release deve ser homologada, e possíveis correções serão feitas diretamente nela. Após a homologação da release branch é feito o merge desta para master e develop.

Caso seja encontrado algum bug diretamente em produção, uma hotfix branch pode ser criada, uma branch com curto período de duração, aberta de master, aonde o objetivo do desenvolvedor é corrigir o bug com o menor impacto possível. Após o desenvolvimento e os testes, é feito o merge desta branch diretamente para o master e uma nova versão é publicada em produção.

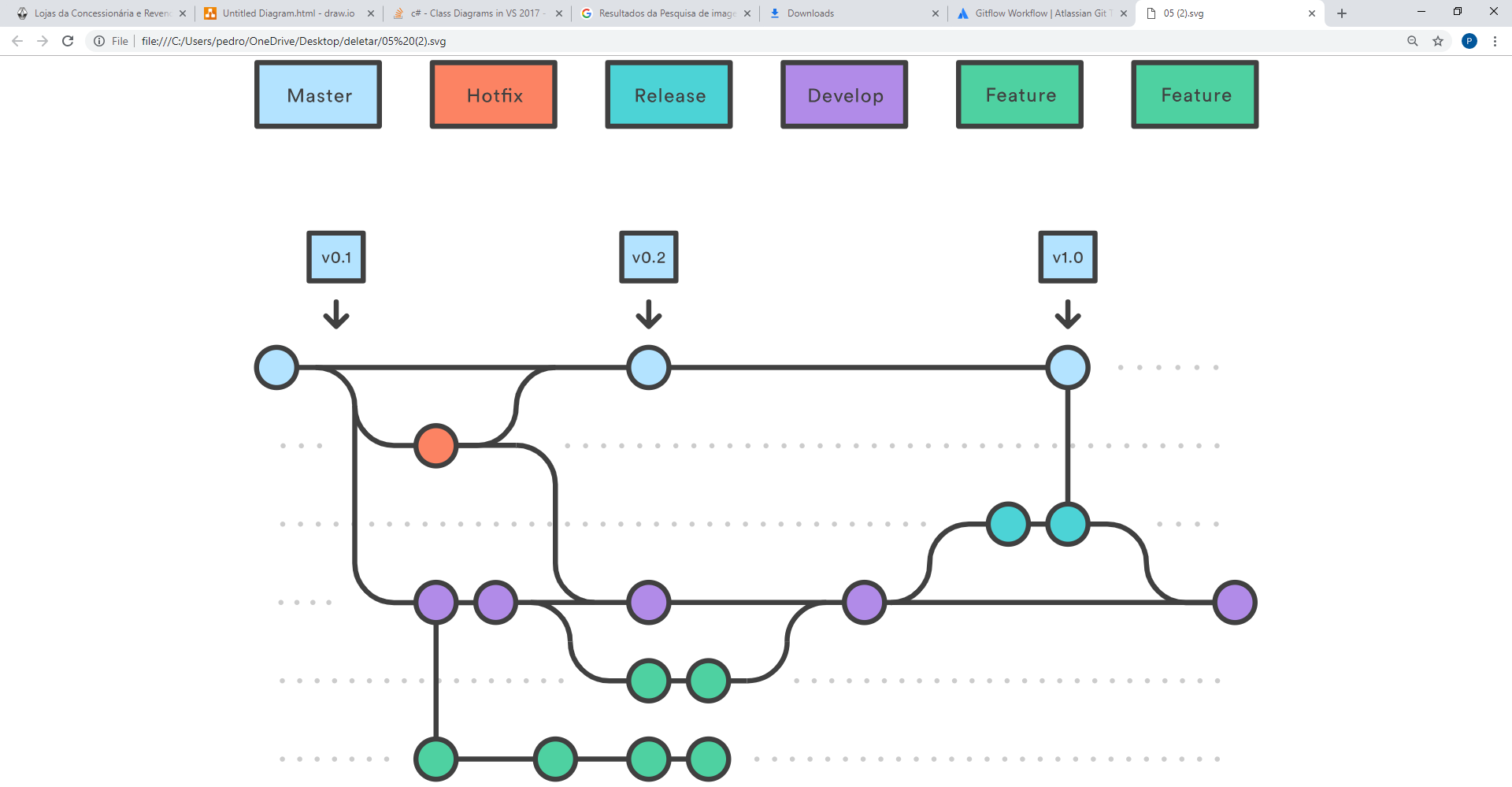


Figura 4 - Estratégia de Branchs GitFlow  
<https://br.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

### 

### 2.1.6. GIT HUB

O GitHub é uma plataforma de armazenamento de repositórios Git na nuvem. Com uma comunidade enorme de desenvolvedores, essa plataforma hospeda repositórios de inúmeras linguagens e tecnologias. O serviço é gratuito para repositórios públicos, o que estimula a colaboração no desenvolvimento de software.

Todo o repositório GIT da Cápsula do Tempo está publicado no GitHub, e pode ser consultado a partir do endereço <https://github.com/Tonis2222/CapsulaDoTempo>.

### 2.1.7. DOCKER HUB

O DockerHub é uma plataforma online de armazenamento de imagens Docker. Por meio dela o desenvolvedor pode hospedar e recuperar suas imagens de qualquer lugar. O serviço é gratuito para imagens públicas, o que estimula a colaboração de compartilhamento de conhecimento e tecnologias.

### 2.1.8. MONGO DB

O MongoDB é um banco de dados não relacional altamente escalável de código aberto e gratuito. Por não ser um banco SQL, o desenvolvedor não precisa criar seus esquemas antes de começar a utilizar o MongoDB, o que acelera muito o processo de desenvolvimento de software.

# 3. TRABALHOS RELACIONADOS

## 3.1 DONT PAD

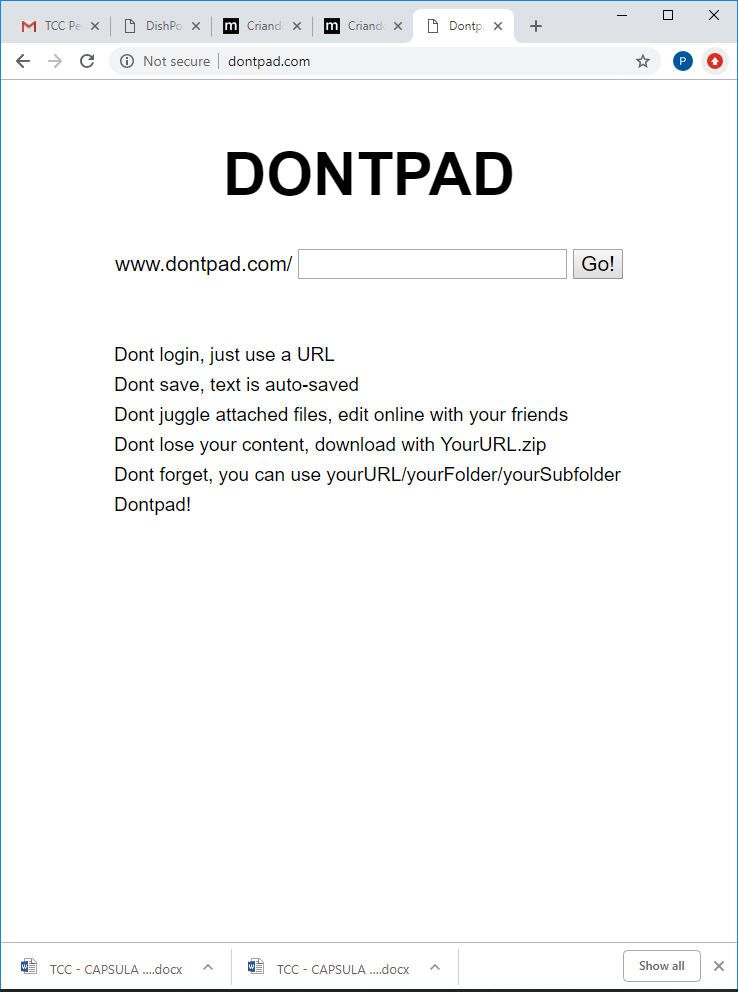


Figura 5 - Interface DONTPAD

O DontPad é um aplicativo Web de armazenamento e compartilhamento de texto. A forma de acesso a um texto pré-armazenado é pelo nome e este vai na url. O DontPad usa a url para identificar um texto ou criar um novo assim que o usuário entra no aplicativo. Sem controle de acesso, login ou qualquer outro controle, o aplicativo permite o compartilhamento rápido de textos em urls com fácil acesso.

A cápsula do tempo utiliza o mesmo conceito de acesso por nome e permite que qualquer usuário seja capaz de acessar qualquer cápsula existente, ver o seu conteúdo caso esta esteja aberta ou criar uma nova caso não exista.

O DontPad tem uma funcionalidade de sugestão de textos com o nome próximo ao que foi buscado. Esta funcionalidade não existirá na cápsula do tempo, pois um requisito importante do aplicativo é que as cápsulas não sejam indexadas em nenhum lugar.

<http://dontpad.com>

## 3.2 SNAPCHAT

O SnapChat é um aplicativo de compartilhamento de imagens, no qual um usuário compartilha uma imagem com outras pessoas e determina um tempo de duração da imagem. Após o tempo pré-determinado, a imagem é excluída e não pode mais ser visualizada.

O aplicativo de cápsula do tempo utiliza o mesmo conceito de volatilidade do Snapchat. Assim, uma cápsula ficará no ar por um determinado período de tempo e, após ser descartada, outro usuário poderá criar uma nova cápsula com o mesmo nome. A maior diferença entre os dois aplicativos é que na cápsula do tempo o usuário pode determinar o momento exato de disponibilização do conteúdo.

<https://www.snapchat.com>

## 3.3 PASTEBIN

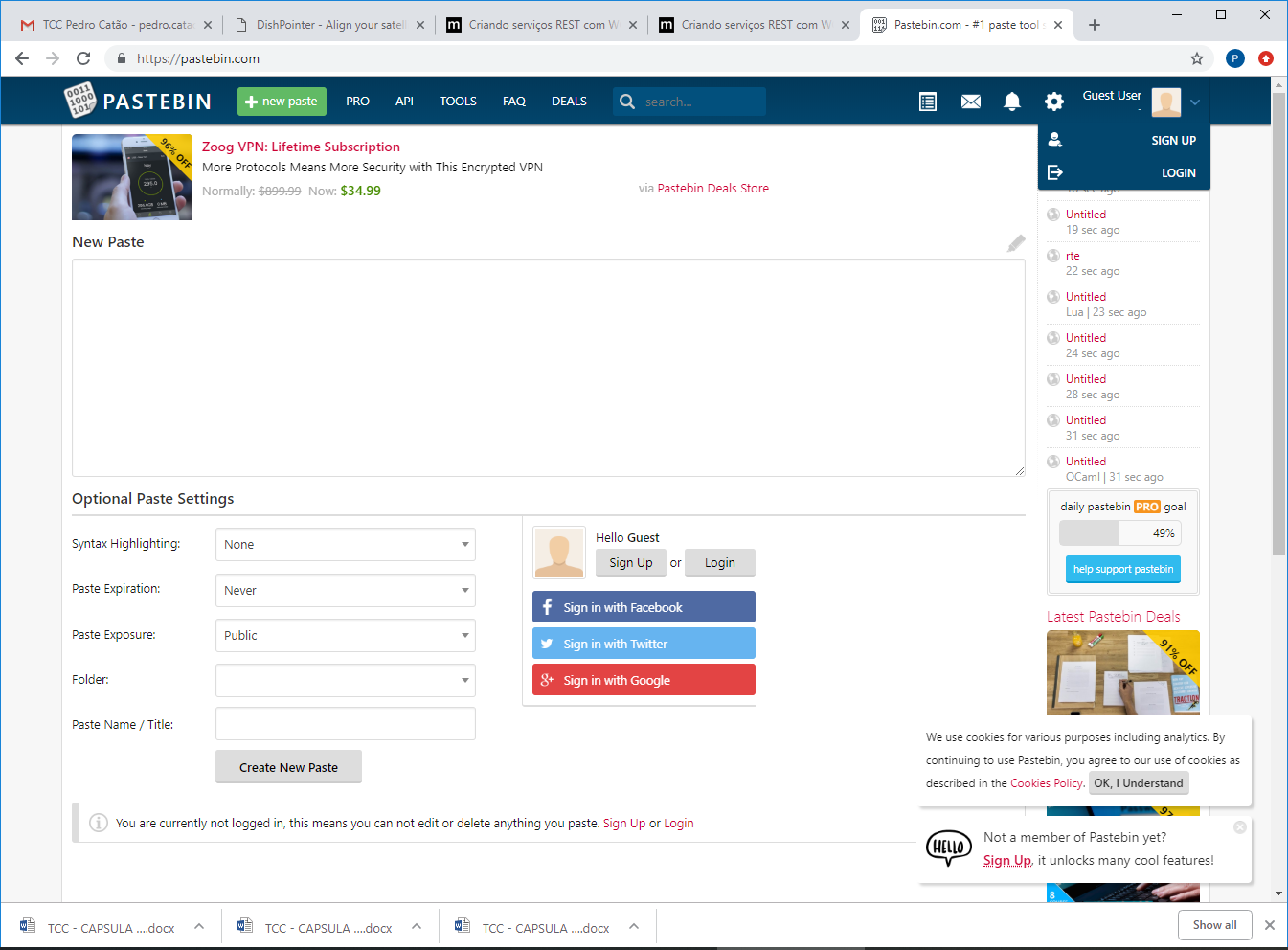


Figura 6 - Interface PASTEBIN

O Pastebin é um dos mais conhecidos aplicativos de compartilhamento de conteúdo, sua interface é cheia e possui muitas configurações diferentes para a disponibilização do conteúdo. O aplicativo permite até colocar uma data de expiração do conteúdo, mas não permite definir uma data para disponibilização do mesmo. A URL do conteúdo é gerada de forma aleatória, sendo difícil de ser memorizada e compartilhada.

A cápsula do tempo simplifica o processo de criação e compartilhamento do conteúdo, possibilitando ainda o compartilhamento de imagem e a programação de quando o conteúdo passará a estar disponível para consumo. Outra diferença é a edição de conteúdo, pois no Pastebin o criador de um conteúdo pode editá-lo a qualquer momento, mesmo que a disponibilização esteja programada para uma data futura, o que não garante que o conteúdo postado originalmente será aquele exibido na data programada. Já a cápsula do tempo não permite que ninguém edite uma cápsula fechada, garantindo assim que aquele conteúdo postado originalmente será disponibilizado sem modificações.

<https://pastebin.com/>

# 4. SISTEMA

Neste capítulo, será apresentado o atual trabalho em dois subcapítulos: no primeiro, veremos a visão geral do sistema, explicando de forma sucinta seus módulos e funcionalidades; no segundo, serão mostradas as arquiteturas que compõem o sistema, sua topologia e a contextualização técnica.

## 4.1. VISÃO GERAL

O presente projeto tem como objetivo funcionar como uma cápsula do tempo, na qual o usuário insere uma imagem, um comentário e uma data de abertura da cápsula.

Enquanto a data de abertura da cápsula não é alcançada, a cápsula fica fechada e seu conteúdo não pode ser consultado. Toda cápsula ficará aberta durante um período igual ao que ficou fechada e após este tempo a capsula será destruída. Uma vez criada, a cápsula do tempo não pode ser editada. Para garantir que um conteúdo seja exatamente aquele postado originalmente, a edição não pode ser exequível.

Atualmente, o aplicativo não exige nenhum tipo de autenticação para postar e consultar cápsulas e as cápsulas ficam sempre na mesma URL. Caso não exista nenhuma cápsula na URL informada, o app permitirá ao usuário a criação de uma cápsula nesta URL.

A aplicação se divide basicamente em duas partes: a API, que é responsável por expor as funcionalidades do sistema para ser consumido por qualquer UI, e a parte de UI, que é a camada de apresentação, responsável por exibir no browser do usuário as cápsulas expostas pela API de forma amigável

### 4.1.1. API

Esta parte do sistema é responsável por expor as funcionalidades do sistema e é acessível apenas por outros sistemas. A API é acessada a partir dos verbos HTTP, conforme descritos no referencial teórico. O verbo GET é responsável por informar se a cápsula existe ou não e se a mesma está aberta ou fechada. E o verbo POST é responsável por incluir uma nova cápsula na base de cápsulas da aplicação.

### 4.1.2. WEB UI

Esta parte do sistema é a responsável por mostrar no browser do usuário as telas do sistema e permite ao usuário uma navegação amigável.

O sistema funciona da seguinte forma: o usuário deve entrar na URL da cápsula que ele quer criar ou consultar. Exemplo: <http://capsulas.com.br/minhaCapsula>, no qual a parte da URL “minhaCapsula” é o nome da capsula a ser procurada.

Caso o sistema não encontre a cápsula com o nome “minhaCapsula”, será apresentada para o usuário uma tela de criação de cápsulas. Caso o usuário submeta uma cápsula com uma imagem, um comentário e uma data de abertura futura, o sistema armazenará a capsula e iniciará na tela uma contagem regressiva até a abertura da mesma.

Caso a cápsula “minhaCapsula” exista e esteja fechada, é apresentada na tela uma contagem regressiva.

Caso a cápsula “minhaCapsula” exista e esteja aberta, é apresentada uma tela com a imagem e o comentário da cápsula que foi armazenada previamente.

Caso a cápsula “minhaCapsula” exista, mas esteja expirada, o sistema irá excluir a capsula armazenada e mostrará ao usuário a tela de criação de cápsula.

## 4.2. ARQUITETURA

A arquitetura de software tem como definição o compartilhamento do entendimento do projeto. Esse entendimento tange à divisão do sistema em componentes e à forma pela qual esses componentes devem interagir com sua interface. No geral, o conhecimento deste funcionamento por todos os seus desenvolvedores é o que se pode chamar de arquitetura (Martin Fowler, Who need an architeture?).

O sistema é dividido dentro de uma arquitetura em comum conhecida por Arquitetura em Camadas. Essa arquitetura ganhou notoriedade, na década de 90, a partir de sistemas cliente/servidor onde o cliente possuía a interface e os códigos de aplicação e o servidor dispunha de um banco relacional que era acessado pelo cliente. Com essa arquitetura denominada por Duas Camadas, era possível facilitar o desenvolvimento de aplicações que realizavam diversas chamadas de dados.

Com o surgimento da lógica de domínio, os sistemas cliente/servidor acabam perdendo a eficácia já que, além de se tornar deselegante, manter regras de negócio na interface do usuário traz ineficácia na manutenção e produção dos códigos, bem como, fatalmente, induzem sua duplicação. Com isso, o aumento da popularidade da orientação a objeto foi favorecida, por ter como resposta o desenvolvimento em três camadas, que possui um esquema parecido com o apresentado nesse projeto, à exceção da camada de serviço, na qual as demais camadas apresentam funções semelhantes às descritas nessa abordagem (Martin Fowler, David Rice, Matthew Foemmel,Edward Hieatt, Robert Mee, Randy Stafford - Patterns of Enterprise Application Architecture). Suas camadas se dividem da seguinte forma: a API abriga a camada de apresentação, a lógica de domínio se encontra na camada de Domínio e, por fim, a camada de infraestrutura é onde se localiza o Banco de Dados.

Com o passar do tempo, alguns problemas de arquitetura e manutenibilidade foram se tornando frequentes e algumas soluções-padrão foram sendo utilizadas em alguns lugares. Para criar um conjunto de princípios e padrões, o livro “Domain-Driven Design: Tackling Complexity in the Heart of Software” foi escrito. Este livro levou muitos desenvolvedores a pensar de forma diferente. O livro prega que o software deve possuir toda a sua lógica de negócio isolada no núcleo do software, facilitando assim a manutenção e organização do código.

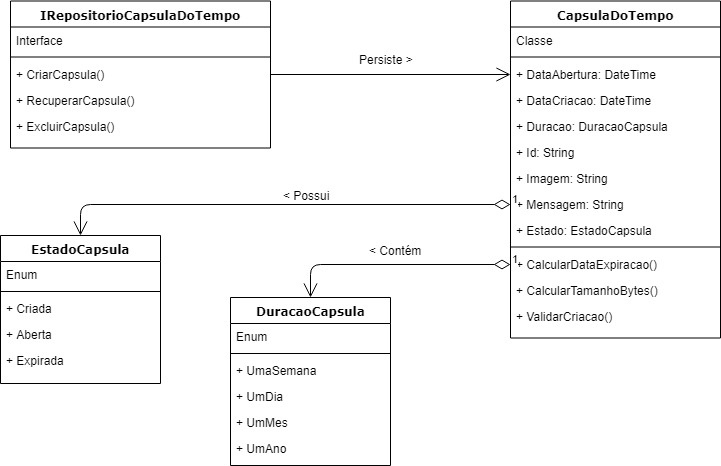


Figura 7 - Diagrama de classes - Domain model

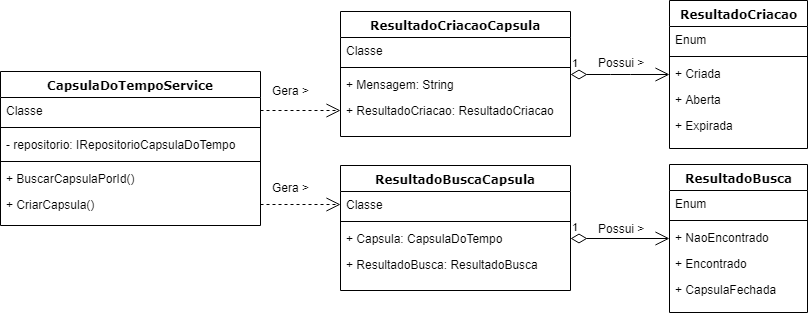


Figura 8 - Diagrama de classes - Domain service

### 4.2.1. CAMADA DE INFRAESTRUTURA - PERSISTÊNCIA OU CAMADA DE DADOS

Por se tratar de um banco de dados não relacional, não foi preciso modelar suas tabelas e relacionamentos, apenas é necessário armazenar os objetos e o MongoDB os guarda e depois sabe remontá-los. Por ser uma aplicação distribuída na nuvem, a opção de um banco de dados na nuvem também foi escolhida. O Azure Cosmos DB foi escolhido por ser um banco na nuvem replicado em diferentes geolocalizações, com redundância e suporte da Azure.

A Interface de como a camada de banco de dados deve se comportar está definida no domínio e cabe à camada de infraestrutura implementar esta interface e fazer funcionar.

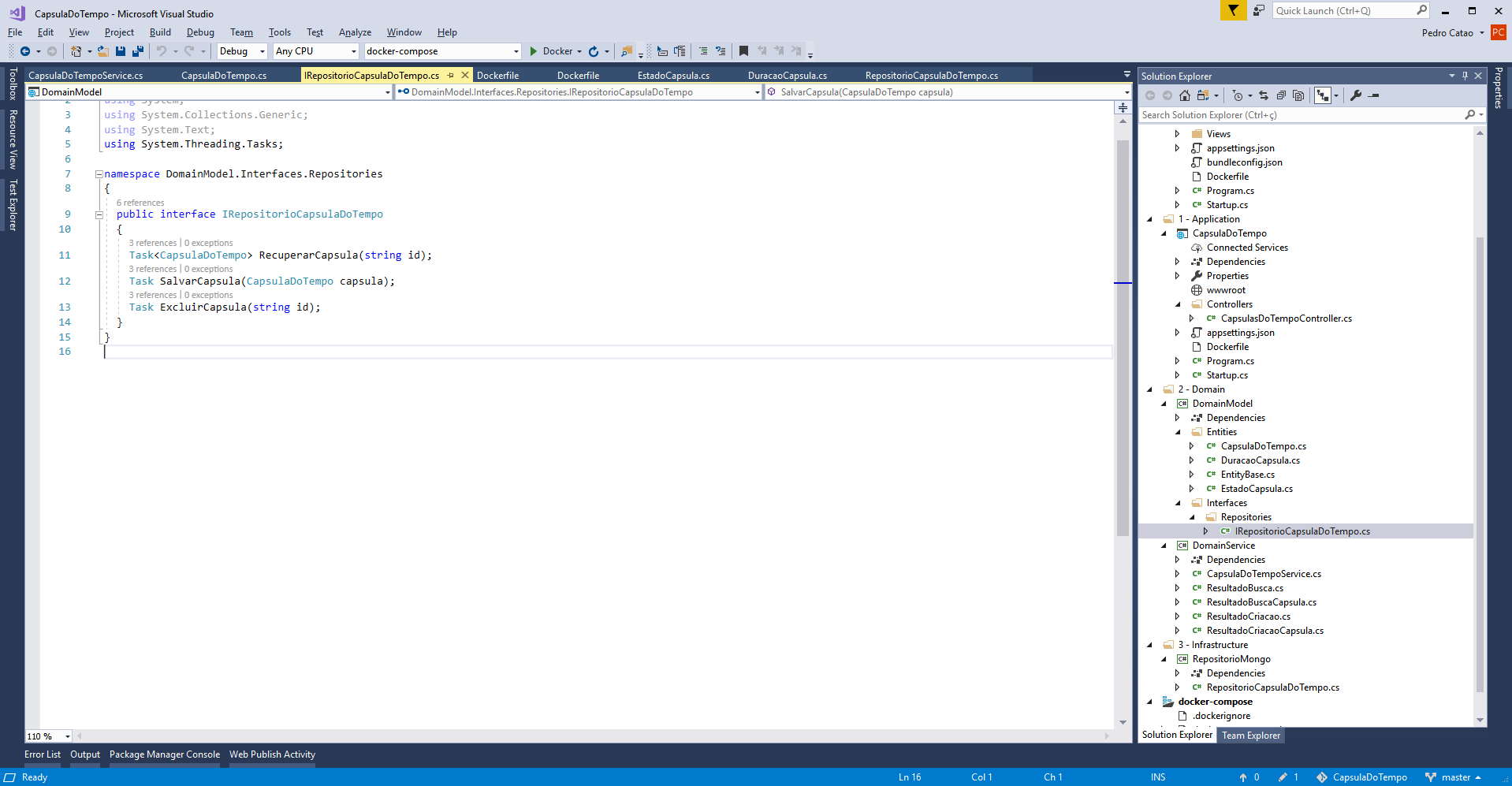


Figura 9 - Interface do repositório de Cápsula do Tempo no Domínio

O modelo de dados do sistema é bem reduzido e contém apenas uma entidade, a CapsulaDoTempo. O acesso ao MongoDB fica concentrado na implementação da interface mostrada acima. Na classe RepositorioCapsulaDoTempo, o acesso é feito pelo pacote MongoDB.Driver, que fornece uma serie de recursos para a comunicação com o MongoDB.

Por ser um banco de dados NoSQL, as queries não foram escritas em SQL, pois no próprio pacote de conexão com o banco existe uma serie de métodos que permitem o acesso aos dados.

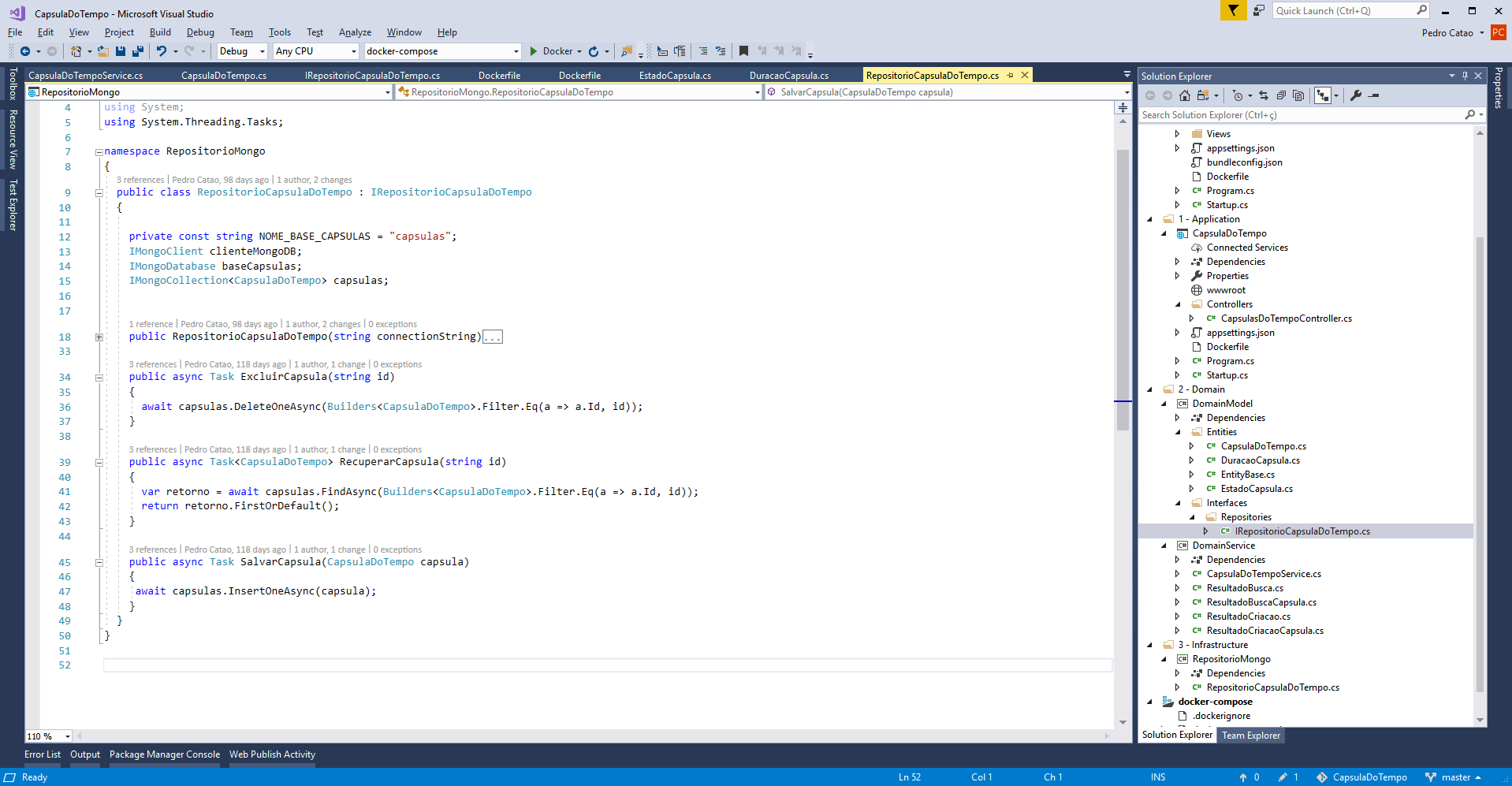


Figura 10 - Implementação do Repositório de Cápsula do Tempo

*.*

### 4.2.2. CAMADA DE DOMÍNIO

A Camada de Domínio é onde os conceitos fundamentais e as regras do negócio são representados. Quando há necessidade da utilização do Modelo de Domínio, ele é considerado o "core" do sistema, o núcleo, influenciando para que as decisões do desenvolvimento e o crescimento das outras camadas dependam da forma que este foi criado (Beyond Software Architecture). O Modelo deste projeto é pequeno e possui poucas regras de negócio, mas estas estão isoladas e apresentam fácil manutenibilidade.

Utilizando o DDD, toda a lógica do sistema se concentra na camada de domínio e esta é referenciada por todas as demais camadas, mas não possui referência a nenhuma. Assim, a camada de domínio se encontra no “meio” do software. Utilizando outros conceitos pregados pelo DDD, como o conceito SOLID, temos os benefícios da organização e da fácil manutenção do código, mesmo que seja um código que sofreu muitas alterações ao longo do tempo.

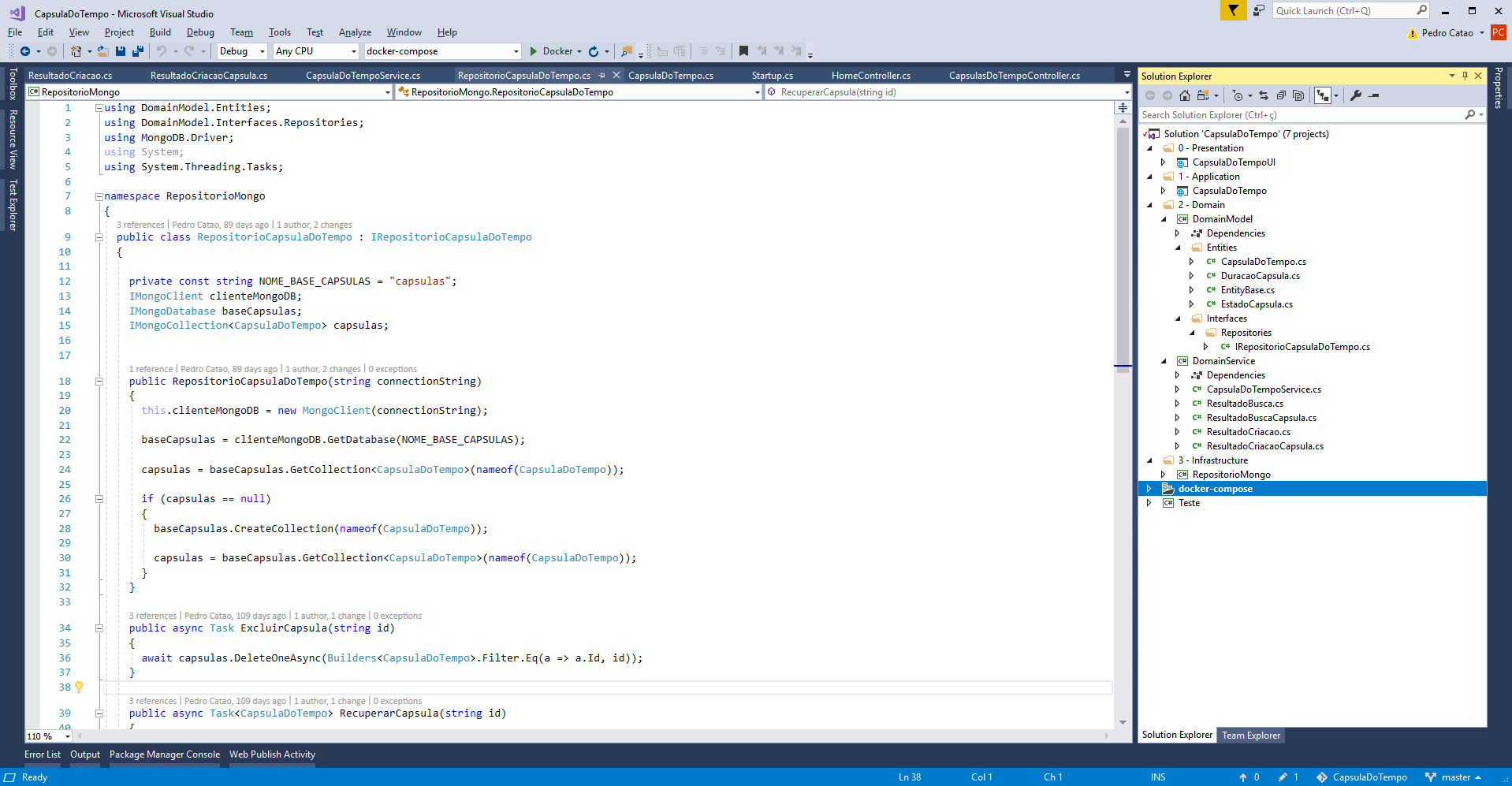


Figura 11 - Estrutura do código

### 4.2.3. CASOS DE USO

Os casos de uso identificados no aplicativo são os seguintes:

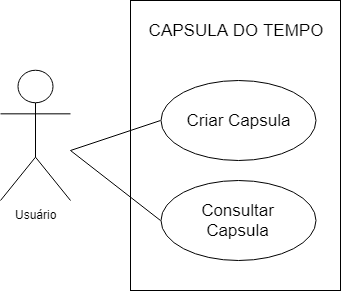


Figura 12 - Casos de Uso

Os casos de uso estão detalhados no Anexo 1 – Casos de Uso. No código fonte, a lógica de negócio dos dois casos de uso são implementados na camada de domínio, na classe CapsulaDoTempoService.cs, que possui dois métodos principais com nomes autossugestivos: BuscarCapsulaPorId e CriarCapsula. Nessa classe, está a implementação das regras de negócio referentes aos casos de uso do sistema. Em um sistema mais complexo, as regras de negócio dos casos de uso estariam divididas em mais classes, mas no domínio da cápsula do tempo apenas uma classe contém a lógica de todos os casos de uso do sistema.

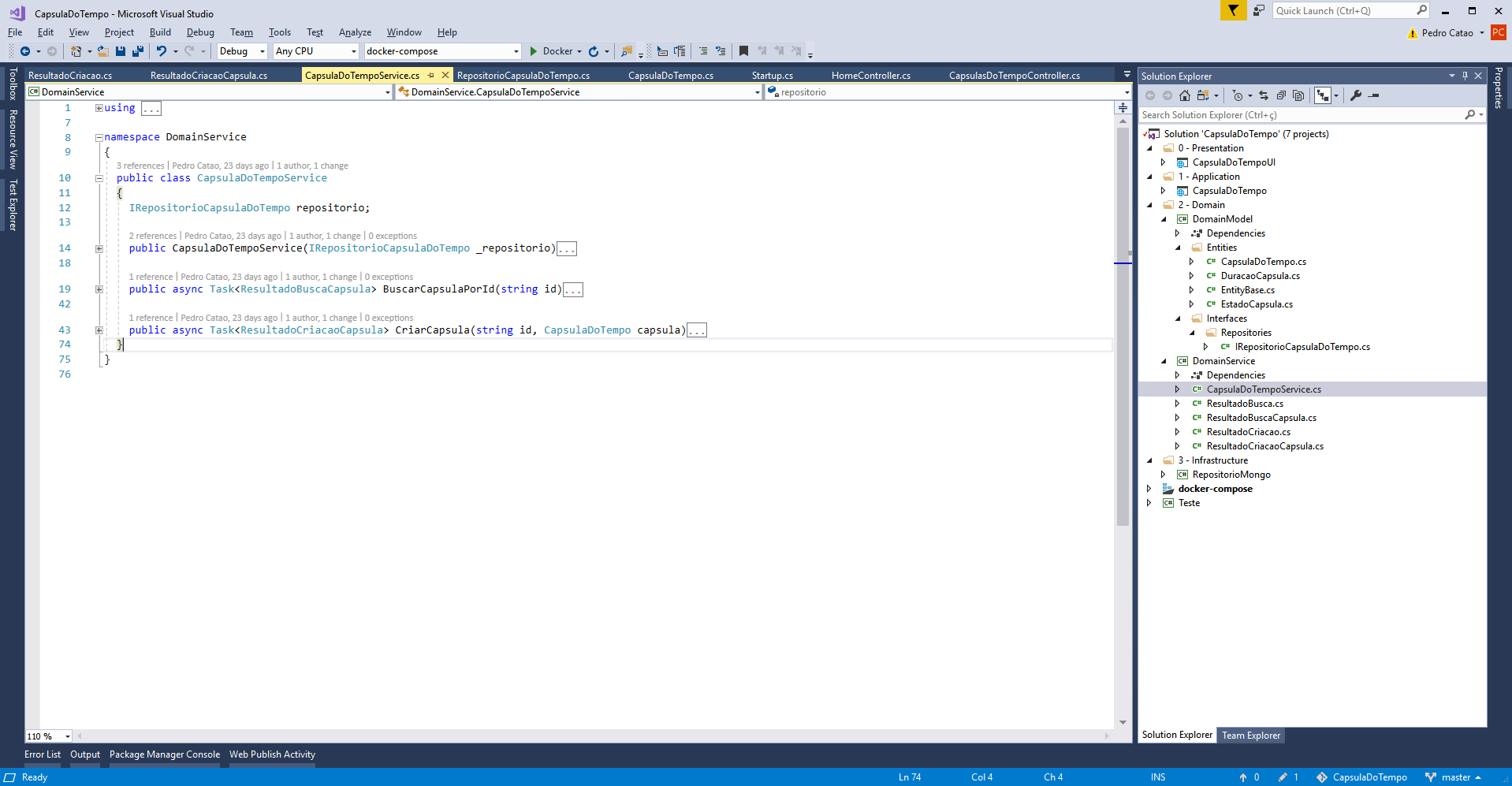


Figura 13 - Classe CapsulaDoTempoService

Os diagramas de sequência dos casos de uso podem ser vistos no Anexo II – Diagramas de Sequência. Para facilitar a compreensão, a camada de apresentação foi ocultada e a parte mostrada é apenas das camadas de aplicação, domínio e infraestrutura.

No diagrama de sequência “Buscar Capsula do Tempo aberta”, podemos ver a interação entre as camadas. Partimos de uma camada mais externa, que é o controller na camada de aplicação, que por sua vez chama o serviço de domínio CapsulaDoTempoService. Esse faz uma chamada indireta para a camada de infraestrutura utilizando a interface implementada, definida pelo domínio “IRepositorioCapsulaDoTempo”. Encontrando uma cápsula do tempo, a camada de infra responde à camada de domínio, que por sua vez responde à camada de aplicação, que finalmente responde ao usuário sobre a cápsula do tempo procurada.

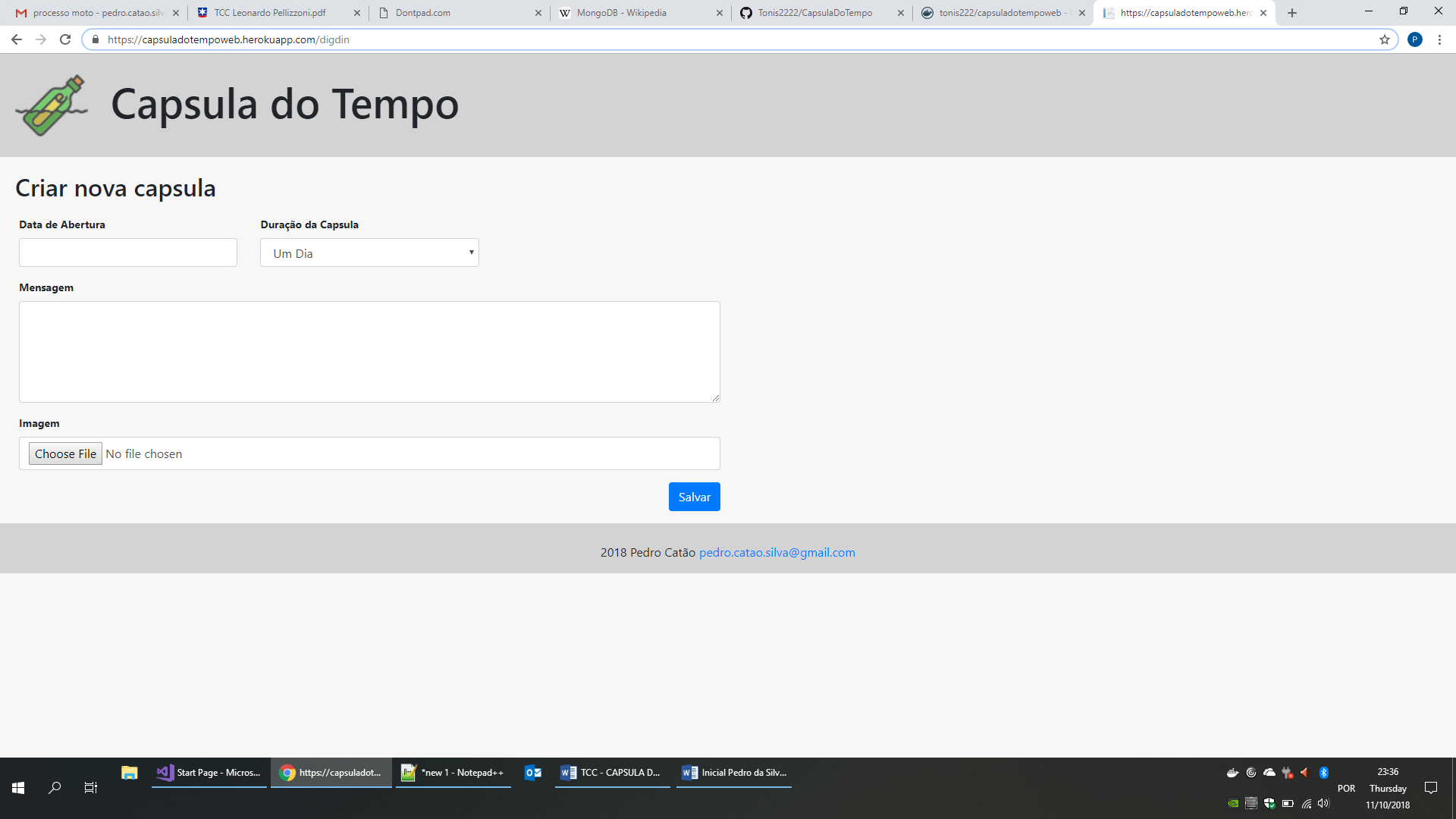
Os demais diagramas de sequência de busca mostram outros resultados da busca de cápsulas, alterando apenas a resposta HTTP.

O diagrama “Criar Capsula do Tempo” possui o fluxo mais complexo do sistema.Nele, o serviço de domínio CapsulaDoTempoService tem uma série de passos a fazer. Após construir o objeto CapsulaDoTempo, este faz o set das propriedades Id e DataCriacao. O próximo passo é validar a criação da cápsula, depois chamar o repositório para garantir que esta cápsula ainda não exista e, por fim, criar a cápsula informada pelo usuário.

# 5. CONCLUSÃO

O aplicativo proposto foi desenvolvido e o código está disponível no endereço <https://github.com/Tonis2222/CapsulaDoTempo> . As imagens Docker foram publicadas nos endereços <https://hub.docker.com/r/tonis222/capsuladotempo/> e <https://hub.docker.com/r/tonis222/capsuladotempoweb/> . Para provar o conceito do desenvolvimento independente de ambiente, a mesma imagem foi publicada nas nuvens da Azure e Heroku e estão disponíveis nos endereços <http://capsuladotempoweb.azurewebsites.net/> e <https://capsuladotempoweb.herokuapp.com/> .

A interface web pode ser vista na figura a seguir:



Foi proposto pelo orientador que, para facilitar o entendimento sobre a duração de uma cápsula, a lógica fosse alterada e as capsulas tivessem durações pré-configuradas. Esta foi a única alteração no funcionamento descrito nos objetivos.

Um desafio encontrado durante as fases de teste foi o custo da nuvem, pois o Azure Cosmos tem um custo fixo por tempo de operação e o container na Azure um custo por requisição, o que consumiu os créditos da assinatura MSDN. O Heroku possui um plano gratuito com o consumo de memória e CPU limitados, mas que para os testes do protótipo foram o suficiente.

O resultado obtido foi satisfatório, o aplicativo está publicado na nuvem, em funcionamento e com todas as regras previstas funcionando. O código está publicado no GitHub e pode ser acessado por qualquer pessoa que queira consultar o código e evoluí-lo.

# REFERÊNCIAS BIBLIOGRÁFICAS

Martin Fowler, **Who need an architeture?,** ThoughtWorks 2003

Martin Fowler, David Rice, Matthew Foemmel,Edward Hieatt, Robert Mee, Randy Stafford - **Patterns of Enterprise Application Architecture,** Addison-Wesley Professional 2002

Luke Hohmann, **Beyond Software Architecture: Creating and Sustaining Winning Solutions**, Addison-Wesley Professional 2003

# SUGESTÃO DE TRABALHOS FUTUROS

Pensado desde o início para ser acessado por qualquer UI, a camada de API pode ser acessada por aplicativos móveis. O desenvolvimento desses aplicativos é o próximo passo lógico.

Entre os trabalhos futuros oriundos deste projeto, pode-se pensar em algum tipo de autenticação, como a forma de identificação do usuário que postou a cápsula, com o objetivo de combater cápsulas com conteúdo impróprio (Ex: pornografia infantil).

Também é possível realizar a tradução do sistema para outras línguas, a fim de atingir o máximo possível de usuários.

Outras funcionalidades possíveis de serem criadas são:

- tempo máximo de duração de uma cápsula, para que não sejam criadas cápsulas que nunca serão abertas;

- CAPTCHA para evitar que robôs criem muitas cápsulas em um curto intervalo de tempo;

- sistema de notificações, para que uma cápsula dispare uma notificação para um grupo de pessoas quando esta for aberta;

- telemetria e tratamento de erros;

- e atualização do Bower, que está obsoleto.

# ANEXO I – CASOS DE USO

|  |
| --- |
| **Nome do Caso de Uso:** Criar Cápsula |
| **Sumário**   Permite ao usuário do sistema criar uma Cápsula do Tempo |
| **Ator:** Usuário do Sistem |
| **Precondições**   Nenhuma |
| **Fluxo Principal**   1 – Usuário acessa a URL principal do sistema mais o nome da cápsula desejada ex:(http://capsuladotempoweb.azurewebsites.net/CapsulaCasoDeUso1)   2 – O aplicativo retorna um formulário com os campos  Data de Abertura  Duração da Cápsula  Mensagem  Imagem   3 – O usuário preenche o formulário e clica em “Salvar”   4 – O aplicativo retorna uma tela informando que a Cápsula foi criada e está fechada |
| **Fluxo Alternativo 1 – Uma cápsula com o mesmo nome já existe:**   1 – Usuário acessa a URL principal do sistema mais o nome da cápsula desejada ex:(http://capsuladotempoweb.azurewebsites.net/CapsulaCasoDeUso1)   2 – O aplicativo retorna uma tela informando que a cápsula já existe, e mostra a mensagem “Cápsula fechada” caso a cápsula esteja fechada ou mostra o conteúdo da cápsula caso a mesma esteja aberta |
| **Pós-Condições**   1 – A cápsula está criada e ainda não pode ser aberta |
| **Autor:** Pedro Catão Pereira da Silva  **Data:** 10/12/2018 |

|  |
| --- |
| **Nome do Caso de Uso:** Consultar Cápsula |
| **Sumário**   Permite ao usuário do sistema consultar uma Cápsula do Tempo |
| **Ator:** Usuário do Sistem |
| **Precondições**   Nenhuma |
| **Fluxo Principal**   1 – Usuário acessa a URL principal do sistema mais o nome da cápsula desejada ex:(http://capsuladotempoweb.azurewebsites.net/CapsulaCasoDeUso2)    2 – O aplicativo retorna uma tela informando que a cápsula já existe, e mostra a mensagem “Cápsula fechada” caso a cápsula esteja fechada ou mostra o conteúdo da cápsula caso a mesma esteja aberta |
| **Fluxo Alternativo 1 – Não existe uma cápsula com o nome informado:**   1 – Usuário acessa a URL principal do sistema mais o nome da cápsula desejada ex:(http://capsuladotempoweb.azurewebsites.net/CapsulaCasoDeUso1)   2 – O aplicativo abre um formulário com os campos para a criação de uma cápsula. Vide caso de uso “Criar Cápsula” |
| **Pós-Condições**   Nenhuma |
| **Autor:** Pedro Catão Pereira da Silva  **Data:** 10/12/2018 |

# ANEXO II – DIAGRAMAS DE SEQUÊNCIA

