Padrões tecnológicos

desenvolvimento de sistemas web

**Sumário**

[1. Introdução 3](#_Toc98501309)

[2. Infraestrutura 3](#_Toc98501310)

[3. Arquitetura Física 4](#_Toc98501311)

[4. Arquitetura Lógica 5](#_Toc98501312)

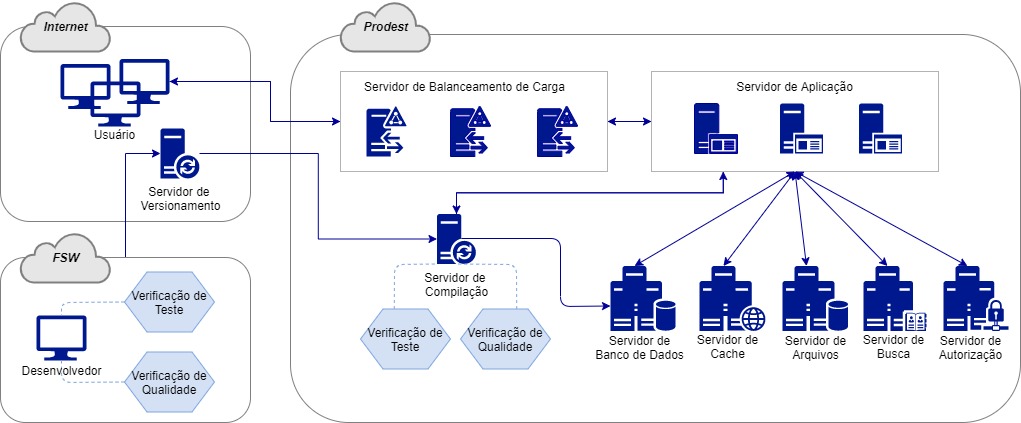
[4.1. Camadas da Arquitetura 5](#_Toc98501313)

# Introdução

O objetivo deste documento é descrever os requisitos tecnológicos recomendados para a construção e verificação de sistemas web do Governo do Estado do Espírito Santo, como condição primária para que possam ser suportados pelo PRODEST. Os requisitos tecnológicos são modelados acerca da análise dos requisitos e incorpora as tecnologias adotadas e as facilidades do ambiente de software.

# Infraestrutura

A seguir é apresentado como se define a infraestrutura disponível no Prodest.



A seguir são apresentadas as especificações dos componentes de infraestrutura recomendados para um produto.

|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Propósito** | **Tecnologia** | **Características** |
| **Servidor de Balanceamento de Carga** | Distribuir as requisições entre os Servidores de Aplicação e fazer offload de SSL/TLS. | IIS – ARR | Servidor configurado sem afinidade de clientes. |
| **Servidor de Aplicação** | Hospedar e executar sites, serviços e aplicativos. | IIS | Servidor web configurado sem sessão, permissão de escrita e permissão de execução, com gerenciamento de ASP.NET. |
| **Servidor de Banco de Dados** | Hospedar e manipular informações do domínio do produto. | MSSQL SERVER | Banco de Dados Relacional. |
| **Servidor de Arquivo** | Armazenar e compartilhar arquivos. | MINIO | Servidor de armazenamento de objetos de forma distribuída. |
| **Servidor de Busca** | Armazenar e analisar informações textuais em arquivos. | ELASTIC SEARCH | Servidor standalone que se comunica com as aplicações via HTTP client e trabalha com arquivos indexados. |
| **Servidor de Cache** | Armazenar cache do usuário. | REDIS | Banco de Dados NOSQL em formato chave-valor com armazenamento de dados em memória. |
| **Servidor de Versionamento** | Versionar o código fonte do produto. | GITLAB | Sistema de controle de versão distribuído mantendo uma cópia do repositório localmente e compartilhando mudanças por meio de um repositório compartilhado. |
| Azure DevOps - Git | Sistema de controle de versão distribuído mantendo uma cópia do repositório localmente e compartilhando mudanças por meio de um repositório compartilhado no Git. |
| **Servidor de Compilação** | Compilar, executar testes e verificar qualidade de código fonte. | Azure DevOps – Agente | Agente executado em plataforma Windows. |
| **Servidor de Autorização** | Controlar o acesso ao produto. | Acesso Cidadão  (API) | API de autenticação de api com protocolo OAuth2. |
| Acesso Cidadão  (Usuário) | API de autenticação de usuário com protocolo OIDC. |

# Arquitetura Física

A seguir são apresentadas as especificações dos componentes da estrutura arquitetural física recomendada para um produto, ou seja, as escolhas de tecnologia definidas para a arquitetura de software.

|  |  |  |  |
| --- | --- | --- | --- |
| **Categoria** | **Propósito** | **Tecnologia** | **Característica** |
| **Plataforma** | Plataforma para desenvolvimento de aplicação C#. | .Net |  |
| **Linguagem Programação** | Linguagem de Programação para codificar as camadas backend. | C# | Habilitar Null check |
| **Framework ORM** | Framework backend para mapeamento objeto relacional. | ENTITY FRAMEWORK CORE | ORM completo |
| DAPPER | Micro ORM |
| **Framework de Interface** | Framework frontend para a interface do produto. | ASP.NET MVC |  |
| BOOTSTRAP | Framework de design com HTML5, CSS3 e JS. |
| **Design de Interface** | Template para o design do frontend do produto, inclusive relatórios. | ADMINLTE | Template de design responsivo com uso de HTML5 e CSS3. |

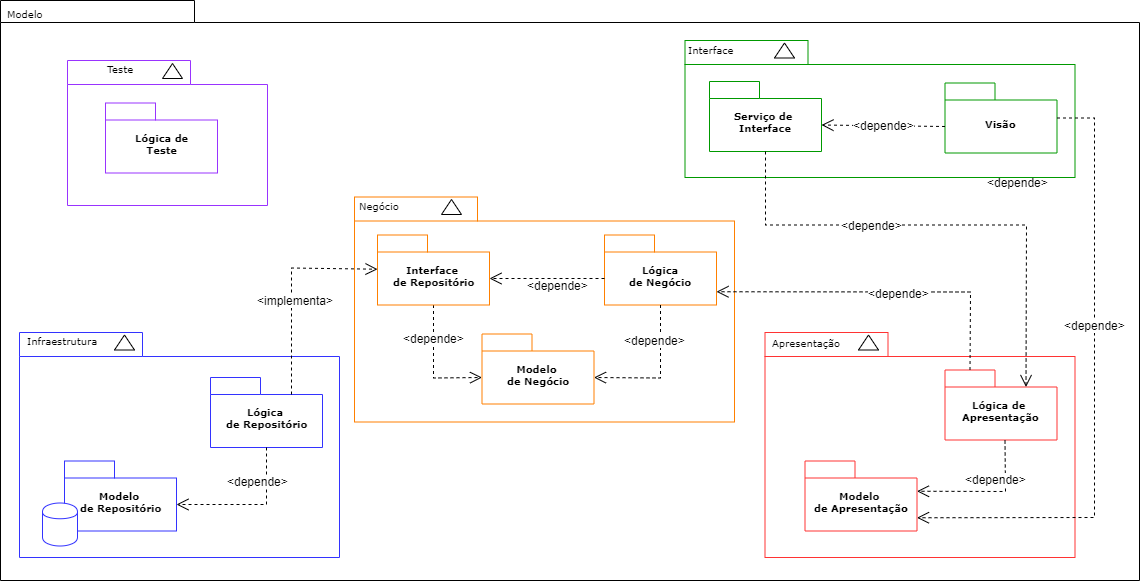
# Arquitetura Lógica

A seguir é apresentado como se define a estrutura arquitetural lógica do produto.

* A arquitetura se baseia em *Onion Architecture* a fim de interagir com as camadas por meio de interface e depender apenas das camadas internas.
* A arquitetura adota o design pattern *Backend for Frontend* a fim de definir uma camada de serviço especifica para comunicação com uma camada de interface específica.

# Camadas da Arquitetura

A seguir são apresentadas as camadas que compõem a arquitetura do produto.



A seguir são apresentadas as camadas da arquitetura.

|  |  |
| --- | --- |
| **Camada** | **Responsabilidade** |
| **Negócio** | Camada core da arquitetura que objetiva representar os objetos do negócio e seu comportamento. Reúne todos os modelos e lógicas do negócio. Essa camada não depende de nenhuma outra camada da arquitetura. |
| **Infraestrutura** | Camada de comunicação com o mundo externo do tipo sistema de arquivos, banco de dados ou sistema externo.  Dado que o Modelo de Negócio pode ser diferente do modelo adequado para o repositório, a Camada de Infraestrutura utilizará um Modelo de Repositório. Assim, a Camada de Infraestrutura irá manipular dados do repositório por meio do Modelo de Repositório e irá transformá-los em modelos compatíveis com a Camada de Negócio. |
| **Apresentação** | Camada que atua como um middleware entre a Camada de Interface e a de Negócio.  Dado que o Modelo de Negócio pode ser diferente do modelo adequado para apresentação na interface, a camada de Interface utilizará um Modelo de Apresentação. Assim, a Camada de Apresentação irá manipular dados da Camada de Interface por meio do Modelo de Apresentação e irá transformá-los em modelos compatíveis com a Camada de Negócio.  Portanto, podem haver diferentes Camadas de Interface que dependem de uma mesma Camada de Apresentação. |
| **Interface** | Camada de comunicação com o cliente, tal como, aplicação web, api, aplicação console ou desktop. |
| **Teste** | Camada que objetiva implementar testes (unitários, integração, etc.) para as outras camadas. |

A seguir são apresentadas as subcamadas de cada camada da arquitetura que será adotada pelo produto.

|  |  |  |
| --- | --- | --- |
| **Camada** | **Subcamada** | **Responsabilidade** |
| **Negócio** | **Modelo de Negócio** | Subcamada que representa os objetos de negócio, ou seja, mantem todos os objetos do domínio. |
| **Lógica de Negócio** | Subcamada que mantem a lógica de negócio (BLL) para o domínio. |
| **Interface de Repositório** | Subcamada que cria uma abstração entre o modelo de negócio e a camada de infraestrutura, ou seja, representa uma interface que será utilizada pela Camada de Negócio.  Portanto, podem haver diferentes Lógicas de Repositório (Camada Infraestrutura) que implementam uma mesma Interface de Repositório (Camada de Negócio). |
| **Infraestrutura** | **Modelo de Repositório** | Subcamada que representa os objetos referentes ao repositório, ou seja, cada repositório mantém seu modelo de objetos. |
| **Lógica de Repositório** | Subcamada que implementa a lógica de acesso a dados da Interface de Repositório e sua persistência.  Portanto, a implementação e a interface são mantidas separadas. Essa subcamada irá manipular dados do repositório por meio do Modelo de Repositório e irá transformá-los em modelos compatíveis com a Camada de Negócio e vice-versa. |
| **Apresentação** | **Modelo de Apresentação** | Subcamada que representa os objetos da apresentação, ou seja, como os dados devem ser apresentados pela Camada de Interface.  Portanto, podem haver diferentes camadas de interfaces que dependem de um mesmo Modelo de Apresentação. |
| **Lógica de Apresentação** | Subcamada que implementa a lógica de apresentação dos dados para a Camada de Interface.  Essa subcamada irá manipular dados da Interface por meio do Modelo de Apresentação e irá transformá-los em modelos compatíveis com a Camada de Negócio e vice-versa. |
| **Interface** | **Visão** | Subcamada que representa a interface visual do software. |
| **Serviço de Interface** | Subcamada que representa a interface do software que adapta solicitações da web. Subcamada que mantem os serviços de interface do software por meio de API e SPI, ou seja, define rotas de acesso disponíveis para requisição a partir da web, da visão ou de outros sistemas. |
| **Teste** | **Lógica de Teste** | Subcamada que implementa a lógica de Teste para as outras camadas. |