# Arquitetura do sistema e tecnologias utilizadas

**Projeto:** Site informativo e de reserva de hotel

**Equipe:**

- Carlos Cauet Ferreira Costa;
- Gustavo Martins.
- Lucas Braga;
- Marcelo Augusto;
- Vanessa Oliveira Lima;

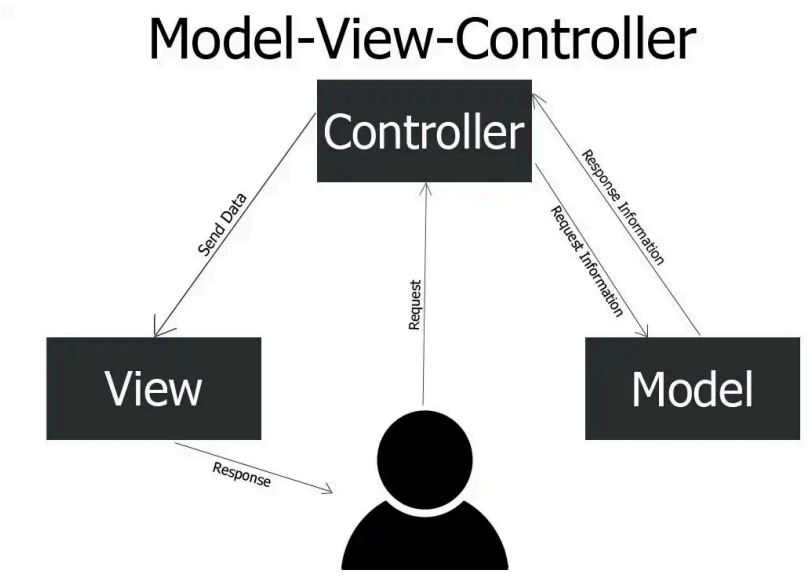
## Introdução

Este documento descreve a arquitetura do sistema e as tecnologias utilizadas para seu desenvolvimento. O sistema será baseado em uma arquitetura MVC (*Model-View-Controller*) e utilizará as tecnologias *Next.js* e *NestJS*. A seguir, detalhamos os módulos, suas responsabilidades e as tecnologias associadas.

## Arquitetura MVC

A arquitetura MVC (*Model-View-Controller*) é um padrão de design que separa a aplicação em três componentes principais:

- *Model* (Modelo): Responsável pela lógica de dados e regras de negócio.
- *View* (Visão): Responsável pela interface do usuário e apresentação dos dados.
- *Controller* (Controlador): Gerencia a comunicação entre o *Model* e o *View*, processando entradas e atualizando o *Model*.



# Tecnologias Utilizadas

## Next.js

***Next.js*** é um framework para React que fornece funcionalidades para construção de interfaces do usuário e renderização do lado do servidor. Ele será utilizado para implementar a camada de *View* do sistema.

### Principais características do *Next.js*:

- **Renderização do lado do servidor** (SSR): Melhora o desempenho e a indexação dos motores de busca.
- **Geração estática** (SSG): Permite criar páginas estáticas durante a construção do projeto.
- **Roteamento automático**: Facilita a criação e gestão das rotas da aplicação.
- **Suporte a API Routes**: Permite a criação de endpoints de API diretamente no projeto Next.js.

### Principais Módulos no Next.js:

- **Páginas**: Componentes React que representam as diferentes views da aplicação.
- ***API Routes***: Endpoints para comunicação com o backend.
- **Estilização**: Suporte a CSS Modules, *styled-components*, e outras soluções de estilização.

## NestJS

***NestJS*** é um framework para Node.js que utiliza *TypeScript* e é inspirado em conceitos de *Angular*. Ele será utilizado para implementar a camada de *Model* e *Controller* do sistema.

### Principais características do NestJS:

- **Modularidade:** Permite a construção de aplicações escaláveis e testáveis por meio de módulos bem definidos.
- **Suporte a TypeScript:** Oferece uma experiência de desenvolvimento mais rica e segura.
- **Injeção de Dependências:** Facilita a gestão de dependências e a modularização do código.
- **Integração com bibliotecas populares:** Inclui suporte para bibliotecas como TypeORM, Mongoose, e mais.

### Principais Módulos no NestJS:

- **Controllers:** Gerenciam as requisições e respostas, atuando como intermediários entre o Model e o View.

- **Services:** Contêm a lógica de negócio e manipulação dos dados.
- **Entities:** Definem a estrutura dos dados e interagem com o banco de dados.
- **Repositories:** Abstraem o acesso aos dados, facilitando operações de leitura e escrita.

## Integração entre Next.js e NestJS

A integração entre Next.js e NestJS será feita da seguinte forma:

- **Next.js (Frontend):** Será responsável pela renderização das views e pela comunicação com o backend através de chamadas API.
- **NestJS (Backend):** Gerenciará a lógica de negócio, o armazenamento de dados e as APIs que serão consumidas pelo frontend.


### Fluxo de Dados:

1. O usuário interage com a interface desenvolvida em Next.js.
2. As requisições do frontend são enviadas para os endpoints de API gerenciados pelo NestJS.
3. O NestJS processa essas requisições, interage com o banco de dados e retorna os resultados para o frontend.
4. Next.js atualiza a interface do usuário com os dados recebidos do backend.

## Tecnologias Adicionais

Para complementar as funcionalidades fornecidas por *Next.js* e *NestJS*, podemos utilizar tecnologias adicionais, incluindo:

- **Banco de Dados:** MySQL ou MongoDB.
- **Ferramentas de Construção e Deploy:** Docker para contêineres e ferramentas de CI/CD como GitHub Actions para integração contínua e entrega contínua.

Porém, o foco não se dará nas ferramentas de construção e deploy.

## Conclusão

A combinação de *Next.js* e *NestJS* permite a construção de um sistema robusto e escalável, com uma clara separação entre a camada de apresentação e a lógica de negócio. Enquanto *Next.js* cuida da renderização e interatividade no frontend, o *NestJS* gerencia a lógica de backend e a comunicação com o banco de dados. Essa abordagem modular e a utilização de tecnologias modernas garantem um desenvolvimento eficiente e uma manutenção simplificada.