# Capítulo 1 - Fundamentação teórica

Neste presente capítulo foi abordados argumentos que deram sustentabilidade neste projecto.

## 1.1. GESTÃO DE TURMAS

O aplicativo TurmaPlus visa melhorar a gestão acadêmica e a experiência educacional em escolas de ensino fundamental e médio. Ele oferece soluções para problemas comuns encontrados nesses ambientes, tais como dificuldades na comunicação entre alunos, professores e pais, a necessidade de registrar e acompanhar notas de forma eficiente, a falta de recursos educacionais acessíveis, desorganização nos horários e atividades escolares, e comunicação limitada entre alunos. Ao fornecer uma interface intuitiva, funcionalidades de registro de notas rápidas e precisas, acesso a uma biblioteca digital abrangente, um calendário acadêmico integrado e um chat de grupo para comunicação entre alunos, o TurmaPlus busca resolver esses problemas e melhorar o ambiente educacional para todos os envolvidos.

**2**

*****Mário Isaías Da Silva Coxe– 2023/2024*

*Capítulo 1 – FUNDAMENTAÇÃO TEÓRICA*

## 1.2. Programação

**Programação** é o processo de escrever instruções para computadores executarem tarefas específicas. Essas instruções são escritas em linguagens de programação, que são conjuntos de regras e símbolos compreensíveis tanto para os programadores quanto para os computadores. A programação permite que os computadores automatizem tarefas, processem informações, tomem decisões e executem uma variedade de funções de acordo com as necessidades do usuário. Em essência, a programação é a base da criação de software e aplicativos que utilizamos em nossos dispositivos eletrônicos, desde sistemas operacionais e navegadores web até aplicativos móveis e jogos.

Depois de sabermos o que é programação, é muito bom que tenhamos conhecimento de um tema também bastante importante, **Desenvolvimento de software** é o processo de criação, concepção, design, implementação, teste e manutenção de programas de computador e sistemas de software. Envolve uma série de atividades coordenadas para produzir software funcional que atenda às necessidades específicas dos usuários ou clientes.

O desenvolvimento de software começa com a identificação das necessidades do usuário ou do problema a ser resolvido. Em seguida, os requisitos são analisados e documentados, e um plano é elaborado para o desenvolvimento do software. Durante a fase de design, são criadas as especificações detalhadas do software, incluindo a arquitetura do sistema, interfaces de usuário e fluxos de trabalho.

**3**

*Mário Isaías Da Silva Coxe – 2023/2024*

*Capítulo 1 – FUNDAMENTAÇÃO TEÓRICA*

A implementação envolve a codificação do software com base nas especificações e design previamente definidos. Os programadores utilizam linguagens de programação e ferramentas de desenvolvimento para escrever o código-fonte do software. Em seguida, o software é testado para garantir que funcione conforme o esperado e que atenda aos requisitos do usuário.

****

**Figura 1.1**: Ciclo de vida de desenvolvimento de sistemas.

**Fonte:** Google, 2024

**4**

*Mário Isaías Da Silva Coxe – 2023/2024*

*Capítulo 1 – FUNDAMENTAÇÃO TEÓRICA*

## 1.3. multi-tenancy.

Mult-Tenancy é um estilo de arquitetura onde você tem uma aplicação centralizada que atende a vários clientes. Neste caso, partindo do Inglês tenant, “clientes” significam locatários ou inquilinos, ou seja, Multi-tenancy (ou multi-tenant) é um termo utilizado em plataformas SAAS, plataformas que oferecem Software Como Serviço, onde, na maioria das vezes os tenants são clientes corporativos.

Essa abordagem é comum em sistemas de software hospedados na nuvem, onde várias organizações ou clientes usam o mesmo aplicativo, mas operam de forma independente umas das outras. A arquitetura multi-tenant oferece várias vantagens, incluindo a capacidade de escalar facilmente para atender a novos clientes, reduzir custos de infraestrutura, simplificar a implantação e a manutenção do sistema e facilitar a personalização.



**Figura 1.1**: Ciclo de vida de desenvolvimento de sistemas.

**Fonte:** Google, 2024

**5**

*Mário Isaías Da Silva Coxe – 2023/2024*

*Capítulo 1 – FUNDAMENTAÇÃO TEÓRICA*

Nesse modelo, cada cliente tem sua própria base de dados dedicada, que é isolada das bases de dados dos outros clientes. Isso permite que cada cliente tenha controle total sobre seus próprios dados e recursos, garantindo a segurança e a privacidade das informações. Essa abordagem é comumente usada em sistemas multi-tenant, nos quais várias organizações compartilham a mesma instância de aplicação, mas têm seus dados armazenados separadamente em bases de dados segregadas. Isso permite uma escalabilidade eficiente e uma fácil personalização para atender às necessidades específicas de cada cliente.



**Figura 1.1**: Ciclo de vida de desenvolvimento de sistemas.

**Fonte:** Google, 2024

A abordagem single database é um modelo de arquitetura de software onde todos os clientes ou inquilinos compartilham um único banco de dados, mas seus dados são isolados logicamente. Isso simplifica a administração e manutenção do sistema, mas pode apresentar desafios de escalabilidade e desempenho. A segurança e privacidade dos dados também são considerações importantes nessa abordagem.

**6**

*Mário Isaías Da Silva Coxe – 2023/2024*

*Capítulo 1 – FUNDAMENTAÇÃO TEÓRICA*

## 1.4. API.

Uma **API** (Application Programming Interface) é um conjunto de definições e protocolos que permite a comunicação entre diferentes softwares. Em termos simples, uma API define como diferentes componentes de software devem interagir entre si. Ela especifica as regras e padrões que os desenvolvedores devem seguir ao acessar e manipular os recursos de um sistema ou serviço.

As **APIs** são utilizadas em uma ampla gama de contextos, incluindo desenvolvimento web, aplicativos móveis, sistemas operacionais, bancos de dados e muito mais. Elas desempenham um papel fundamental na integração de sistemas e na criação de aplicativos que se comunicam entre si.

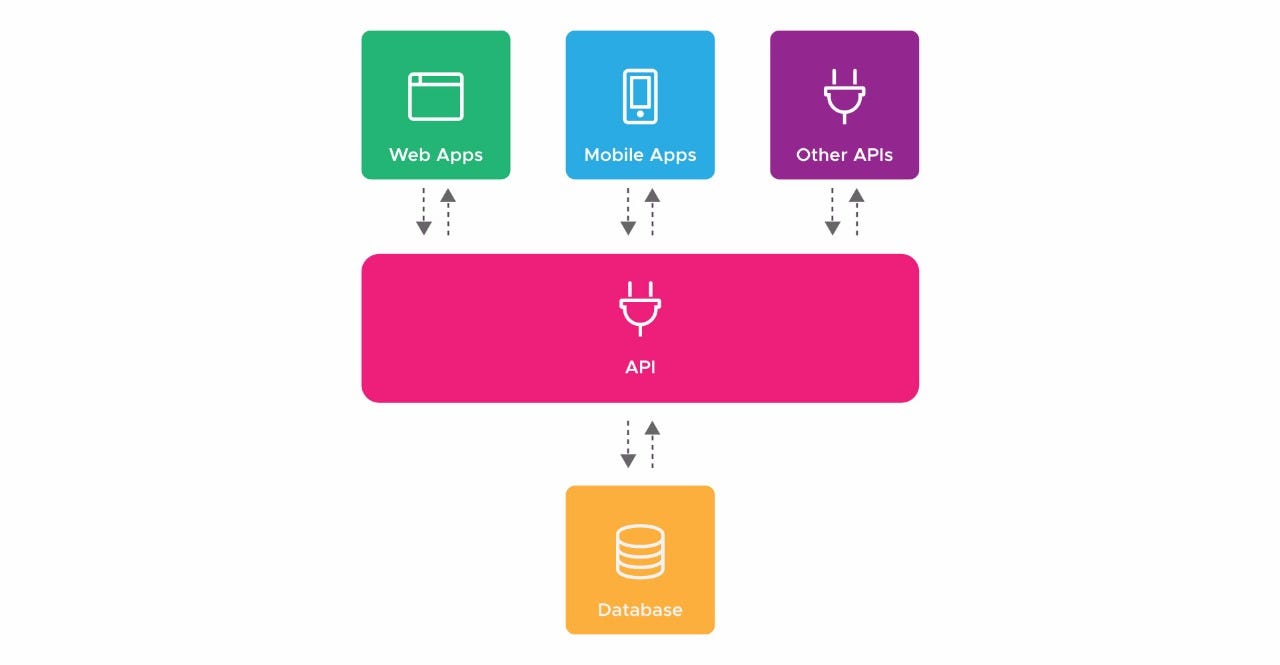
* Uma API REST (Representational State Transfer) é uma interface de programação de aplicativos projetada de acordo com os princípios arquiteturais da REST. Estes princípios incluem:
* Arquitetura Cliente-Servidor: A comunicação é feita entre um cliente e um servidor, onde o cliente faz solicitações e o servidor fornece respostas.
* Comunicação Stateless: Cada solicitação do cliente para o servidor contém todas as informações necessárias para o servidor entender e processar a solicitação. O servidor não mantém informações sobre o estado do cliente entre as solicitações.
* Identificação de Recursos por URIs: Cada recurso acessível pela API tem uma URI única que o identifica de forma única na web.
* Manipulação de Recursos através de Métodos HTTP: Os métodos HTTP (GET, POST, PUT, DELETE) são utilizados de acordo com suas semânticas para realizar operações em recursos. Por exemplo, GET é utilizado para recuperar dados, POST para criar novos recursos, PUT para atualizar recursos existentes, e DELETE para remover recursos.

**7**

*Mário Isaías Da Silva Coxe – 2023/2024*

*Capítulo 2 - METODOLOGIA*

Uma **API REST** é projetada para ser escalável, flexível, e fácil de entender e integrar. Ela segue padrões amplamente reconhecidos da web, como o uso do protocolo HTTP, e é amplamente utilizada para construir serviços web que fornecem acesso a recursos e funcionalidades de u m sistema de forma padronizada e fácil de usar.



**Figura 1.1:** Exemplo do processo de um software que usa API.

**Fonte**: Google, 2024

**9**

*Mário Isaías Da Silva Coxe – 2023/2024*

*Capítulo 2 - METODOLOGIA*

## 1.6. FRAMEWORK.

Um **framework** é um conjunto de ferramentas, bibliotecas, convenções e padrões pré-definidos que fornecem uma estrutura para o desenvolvimento de software. Ele oferece uma base sobre a qual os desenvolvedores podem construir e organizar seus aplicativos de forma consistente e eficiente.

Os **frameworks** são projetados para simplificar o processo de desenvolvimento, fornecendo soluções comuns para problemas recorrentes. Eles geralmente incluem componentes reutilizáveis, como classes, funções e módulos, que podem ser utilizados para implementar funcionalidades específicas sem a necessidade de escrever código do zero.

Existem **frameworks** para uma ampla variedade de finalidades e plataformas, incluindo frameworks web, frameworks para desenvolvimento de aplicativos móveis, frameworks de front-end, frameworks de back-end, frameworks para desenvolvimento de jogos, entre outros.

Alguns exemplos populares de frameworks incluem:

* **Spring (Java):** Framework para desenvolvimento de aplicativos empresariais em Java, bem como criação de APIs.
* **Angular e React (JavaScript):** Frameworks para desenvolvimento de interfaces de usuário web em JavaScript.
* **Express (Node.js):** Framework para desenvolvimento de aplicativa web e APIs em Node.js, bem como criação de APIs.
* **Laravel** é um popular framework de desenvolvimento web em PHP, conhecido por sua elegância, simplicidade e facilidade de uso.

O uso de **frameworks** pode acelerar o desenvolvimento de software, melhorar a qualidade do código, promover a reutilização de código e facilitar a colaboração entre os membros da equipe de desenvolvimento. No entanto, é importante escolher o framework certo para o projeto e entender suas convenções e padrões para maximizar seus benefícios.

**10**

*Mário Isaías Da Silva Coxe – 2023/2024*