**Integrantes: Andrey Masiero, Caio mota, Pedro Eugênio, Rafael Fonseca**

**Nome equipe:T-Miners**

**Tema: Socialização Pessoal**

**Descrição: Aplicativo que busca facilitar o contato entre pessoas que tem os mesmos interesses, através de seus interesses criando grupos e encontros entre elas para que possam formar novas amizades.**

**------------------------------Trabalho----------------------------**

**SOAP**

**O que é?**

O protocolo SOAP, abreviação para Simple Object Access Protocol, é uma especificação para a troca de informação entre sistemas, ou seja, uma especificação de formato de dados para envio de estruturas de dados entre serviços, com um padrão para permitir a interoperabilidade entre eles.

**Características:**

Seu design parte do princípio da utilização de XMLs para a transferência de objetos entre aplicações, e a utilização, como transporte, do protocolo de rede HTTP.

**Vantagens:**

Uma das vantagens do SOAP é o uso de um método de transporte "genérico". Enquanto o REST faz uso de HTTP/HTTPS, o SOAP pode usar qualquer meio de transporte existente para enviar sua requisição, desde SMTP até mesmo JMS (Java Messaging Service).

**REST**

**O que é?**

REST significa Representational State Transfer Em português, Transferência de estado Representacional. Trata-se de uma abstração da arquitetura da Web. Resumidamente, o REST consiste em princípios/regras/constraints que, quando seguidas, permitem a criação de um projeto com interfaces bem definidas. Desta forma, permitindo, por exemplo, que aplicações se comuniquem.

**Características:**

O REST é simples de entender e pode ser adotado em praticamente qualquer cliente ou servidor com suporte a HTTP/HTTPS. Os desenvolvedores que o utilizam citam, como principais vantagens a facilidade no desenvolvimento, o aproveitamento da infraestrutura web existente e um esforço de aprendizado pequeno.

**Vantagens:**

**Separação entre o cliente e servidor**

Uma das vantagens de utilizar o modelo Rest API é a separação entre as aplicações front-end e back-end. Isso é importante para proteger o armazenamento de dados, pois não há o tratamento de regras de negócio, ou seja, é feita apenas a troca de informações seja para recuperar dados, seja para inserir ou deletar novos registros.

**Mais visibilidade, confiabilidade e escalabilidade**

Por ter a separação cliente / servidor, há muito mais facilidade durante o desenvolvimento da aplicação. Isso porque ela pode ser facilmente escalada, já que há não há dificuldade para acoplar recursos. Como cada requisição é feita de maneira única e independente, é possível mudar uma requisição para outro [DNS](https://rockcontent.com/br/blog/dns/), sem que isso interfira na aplicação.

Em outras palavras, a API permite que a aplicação acesse banco de dados de diferentes servidores, o que muitas vezes é importante para o desenvolvimento em grandes aplicações. Portanto, sua utilização garante mais visibilidade e confiabilidade ao utilizar esses recursos.

**Multiplataforma**

As requisições HTTP feitas em uma Rest API retornam dados no formato JSON. Vale ressaltar que existem outros formatos possíveis de retorno, como o XML, entretanto, o JSON é o mais utilizado. Portanto, a maioria dos sites que trabalha sob esse modelo recebe esse formato de dados.

Essa característica é essencial para o desenvolvimento de aplicações multiplataformas. Isso porque, ao receber os dados nesse formato, a camada front-end da aplicação é capaz de fazer o tratamento adequado para a exibição dos resultados de acordo com o tipo de dispositivo utilizado.

**-------------------------TRABALHO 2 -------------------------**

**npm init-y**

**R:** npm usado para publicação de projetos node.js. “-y” quando passado aos comandos NPM diz ao gerador para usar os padrões em vez de fazer perguntas. E também o “-y“significa um sim.

**Package.json**

**R:** responsável por descrever o projeto informar a versão do node e do npm, url projeto dependências de produção e de desenvolvimento dentre outas coisas.

**npm install express**

**R: I**nstala o framework node mais atual e a sua biblioteca subjacente para uma série de outros frameworks no node.

**npm install – save-dev nodemon**

**R:** instala o utilitário que monitora qualquer alteração na sua fonte e reinicia automaticamente o servidor. E o save dev é utilizado para salvar os pacotes necessários para o aplicativo ser executado

**npm run dev**

**R:** basicamente executa o comando que eu defino sobre o nome serve em package.jon.

**-------------------------TRABALHO 3 -------------------------**

A utilização de módulos permite incluir outros arquivos Javascript em sua aplicação.

Módulos são cruciais para construção de aplicações em Node pois eles permitem incluir bibliotecas externas, como bibliotecas de acesso ao banco de dados. também ajudam a organizar seu código em partes separadas com responsabilidades limitadas.

A função require (), recebe um argumento: o nome da biblioteca do core ou o caminho do arquivo do módulo que você quer carregar.

Para um módulo exportar um objeto, ou um construtor, basta adicioná-lo no objeto especial exports. Todas as demais variáveis declaradas dentro do módulo serão variáveis privadas e não serão exportadas.

O Node sempre dá preferência em carregar os módulos do núcleo se seu identificador é passado para a função require(). Por exemplo require (‘http’) irá sempre retornar o módulo http embutido no núcelo do Node, mesmo que tenha um arquivo com o mesmo nome nesta pasta. Caso o nome do arquivo passado para a função require não for encontrado, então o Node irá tentar carregar um arquivo com o nome passado adicionando a extensão .js, .json, e por fim .node`

Um módulo node tem variáveis disponíveis por padrão no escopo de cada módulo, acompanhe a lista abaixo contendo as mais interessantes:

* filename: O nome do arquivo do código que está sendo executado.
* dirname: O nome do diretório que está salvo o script que está sendo executado.
* process: Um objeto que é associado ao presente processo em execução. Além de variáveis, este objeto tem todos como process.exit, process.cwd e process.uptime.
* process.argv: Um **array** contendo os argumentos de linha de comando. O primeiro elemento será node, o segundo elemento será o nome do arquivo JavaScript, e os próximos serão todos os argumentos de linha de comandos adicionais, caso sejam atribuídos.
* process.stdin, process.stout, process.stderr: **Streams** que correspondem à entrada padrão, a saída padrão, e a saída de erro padrão do processo atual.
* process.env: Um objeto contendo as variáveis de ambiente do usuário do processo atual.
* require.main: Quando um arquivo é executado diretamente pelo Node, require.main é atribuído à este module.

**------------------------TRABALHO 4 -------------------------**

FOI REALIZADO O COMMIT DAS 2 CONTROLLERS E DAS 2 ROTAS.