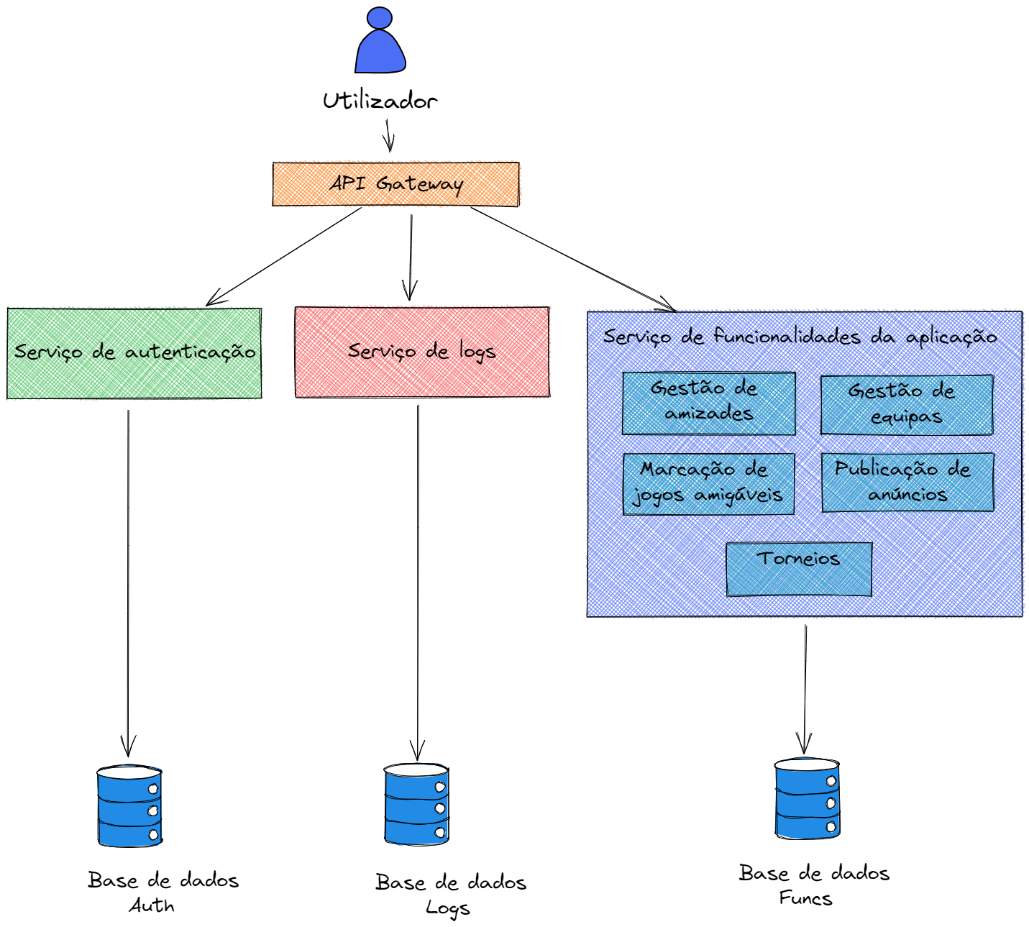
# problema a ser resolvido

-Já esta feito

# abordagem seguida

A abordagem seguida para o desenvolvimento do problema apresentado anteriormente passou pela implementação do modelo arquitetural apresentado na imagem seguinte.



Foram definidos três micro serviços onde cada um tem uma base de dados que armazena informações correspondentes à execução de funções respetivas ao próprio serviço.

Todos os micro serviços foram desenvolvidos utilizando a linguagem nodeJS utilizando a framework express que possibilita a fácil implementação de APIs REST.

Quanto ao nível de estruturação/arquitetura da aplicação foi utilizada a arquitetura *clean* não só porque também foi um requisito para a elaboração deste trabalho prático, mas também porque permite organizar o código de uma maneira onde o número de dependências é bastante reduzido.

## Serviço de autenticação

O serviço de autenticação é o serviço responsável pelo acesso às restantes funcionalidades da aplicação, porque para o acesso a qualquer funcionalidade da aplicação é necessário efetuar login posteriormente de modo a queseja gerado um *token*.

### Registo

O registo na aplicação é efetuado através da rota “…/auth/register” e devem ser fornecidas um conjunto de informações no corpo do pedido.

Esta funcionalidade é uma das que tira proveito da biblioteca Bcryptjs, porque utiliza num registo é sempre necessário fornecer uma palavra-passe e, de modo a proteger esta palavra-passe aquando do seu envio para a base de dados, a mesma é encriptada utilizando os métodos **genSalt** e **hash** para a realização da encriptação.

Após a inserção do novo registo na base de dados é, também, inserido um novo log na base de dados com informações respetivas ao registo como, por exemplo, a hora da realização do mesmo, o nome do utilizador e o código respetivo à funcionalidade que foi executada.

Por fim, existem três valores de retorno possíveis após a execução da funcionalidade de registo:

* 200 – O utilizador foi registado na base de dados com sucesso.
* 401 – O registo falhou, porque o username escolhido para o registo já existe na base de dados.
* 404 - O registo falhou.

- Bibliotecas/Frameworks utilizadas

Express -

Mongoose – Permitiu a estruturação das tabelas do mongoDB.

Dotenv – Permitiu a criação e armazenamento de variáveis de ambiente em vez de serem declaradas no código. Foi útil para guardar informações sensíveis como o *Token* secreto a utilizar na biblioteca Jsonwebtoken e a chave correspondente à utilização do reCAPTCHA.

Router -

Jsonwebtoken – Permitiu a implementação de um serviço de autenticação à base de *tokens* de acesso.

Axios – Equivalente à função *fetch*. Permitiu realizar pedidos ao serviço de *logs.*

Bcrypt – Permitiu implementar um mecanismo de encriptação de informações sensíveis, tais como palavras-passe.

Api-doc – Utilizada para a documentação de todas as funcionalidades da aplicação.

**Testes**

**objetivos atingidos e quais os problemas encontrados**

Inicialmente foi definido que a aplicação ia respeitar a arquitetura MVC (Model-View-Controller), mas com o decorrer das aulas lecionámos nova matéria e, tal como pedido pelo professor, a estrutura da aplicação passou a respeitar a arquitetura *clean.*

A alteração da arquitetura da aplicação foi algo que nos apanhou desprevenidos, porque não só não tínhamos conhecimento qualquer sobre a arquitetura em questão, mas também causou a alteração de todo o código desenvolvido até à data.

Além disso, a aplicação foi desenvolvida utilizando a linguagem nodeJS que não é propriamente uma linguagem ideal para o desenvolvimento de aplicações utilizando este tipo de arquitetura.

**funcionalidades adicionais**

**de que forma é que os componentes da aplicação foram instalados e configurados**

**todas as referências consultadas durante a elaboração do trabalho**