**Rafael Adegas Pires**

**Itaú – Case Técnico – Engenharia de TI**

**12/09/2019 a 18/09/2019**

**repository:** <https://github.com/adegas123/twitter-test>

**Relatório Técnico Final**

**Plano de trabalho**

|  |  |
| --- | --- |
| **Previsto** | **Realizado** |
| Criar uma aplicação Java para coletar as últimas postagens do Twitter para as hashtags: #openbanking, #apifirst, #devops, #cloudfirst, #microservices, #apigateway, #oauth, #swagger, #raml e #openapis. | Criar uma aplicação Java para coletar as últimas postagens do Twitter para as hashtags: #openbanking, #apifirst, #devops, #cloudfirst, #microservices, #apigateway, #oauth, #swagger, #raml e #openapis. |
| Modelar e implementar uma base de dados para armazenar as informações. | Modelar e implementar uma base de dados para armazenar as informações. |
| Coletar os dados e armazenar as mensagens na base de dados | Coletar os dados e armazenar as mensagens na base de dados |
| Sumarizar e gravar os dados para listar os 5 usuários que possuem mais seguidores, o total de postagens agrupadas por hora do dia e o total de postagens para cada um dos termos por idioma/país do usuário que postou. | Sumarizar e gravar os dados para listar os 5 usuários que possuem mais seguidores, o total de postagens agrupadas por hora do dia e o total de postagens para cada um dos termos por idioma/país do usuário que postou. |
| Criar uma API REST em Node.js que permita consumir os dados do passo anterior. | Criada uma API REST em Java que permite consumir os dados do passo anterior. |
| Expor a API no APIgee com utilização de Oauth 2.0. | API não foi exposta no APIgee, mas utiliza Oauth 2.0. |
| Criar uma página em React que chame as APIs e mostre os resultados | Criar uma página em React que chame as APIs e mostre os resultados |

**Justificativa**

**APIgee**

O desenvolvedor optou por não seguir com a disponibilização da API no APIgee pois nunca havia trabalho com um API Gateway e isto demandaria um maior tempo de aprendizado, prejudicando o trabalho.

Por outro lado, o desenvolvedor estava familiarizado com a utilização de Oauth2 para autenticação, portanto foi aplicado na API REST e funcionando com integração pelo Facebook, GitHub e Google.

Esta mudança também ocasionou a alteração na linguagem escolhida para disponibilização dos endpoints que antes seria feita em Node.js e passou a ser feita através da própria aplicação Java.

**Tecnologias**

|  |  |
| --- | --- |
| Backend | - Java 8 |
| - MongoDB |
| - Spring Boot - 2.1.8-RELEASE |
| - Spring Security - managed |
| - Oauth 2.0 - managed |
| - JWT - 0.9.1 |
| - Lombok |
| - Twitter4j - 4.0.7 |
| - Apache Spark Core 2.12 - 2.4.4 |

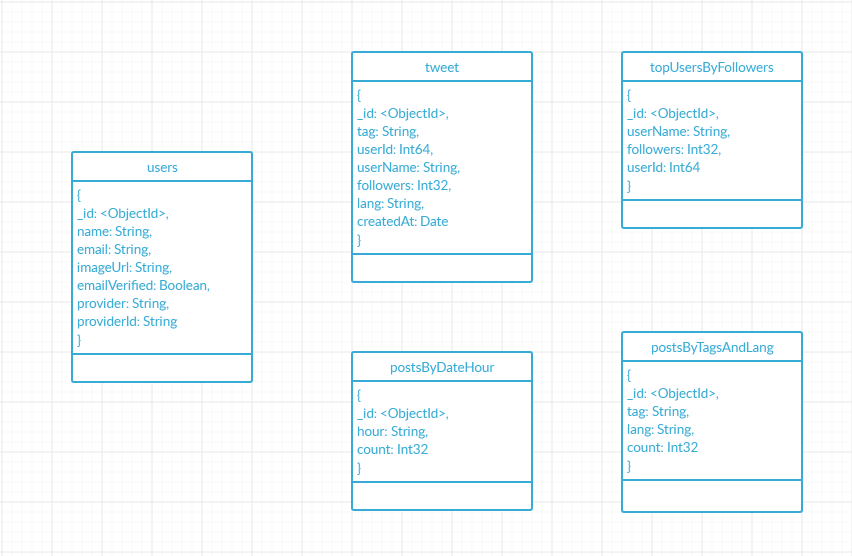
|  |  |
| --- | --- |
| Frontend | - Node.js - v12.7.0 |
| - React - 16.9.0 |
| - Recharts - 1.7.1 |

|  |  |
| --- | --- |
| IDE’s | - Spring Tools Suite 4 |
| - Visual Studio Code |
| Sistema Operacional | - Windows 10 |
| Conteinerização | - Docker for Windows - 2.1.0.1 |
| - Docker Compose - 1.24.1 |

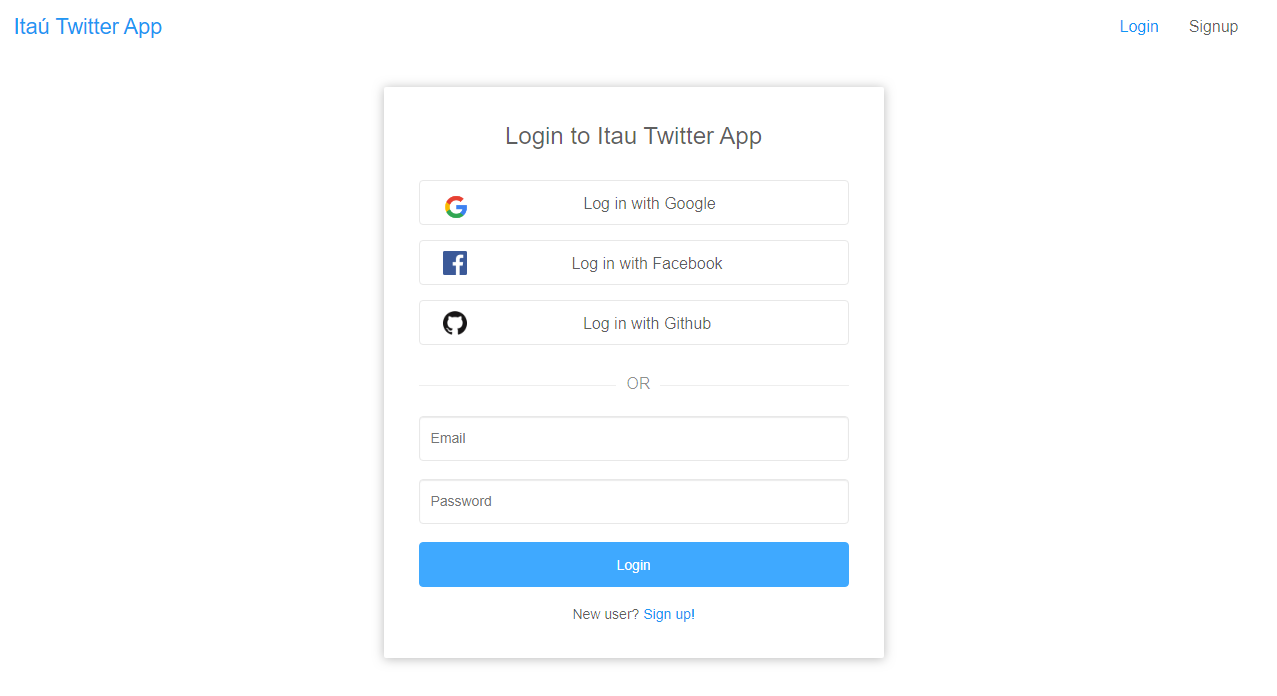
**Arquitetura**

**Modelagem da base de dados**

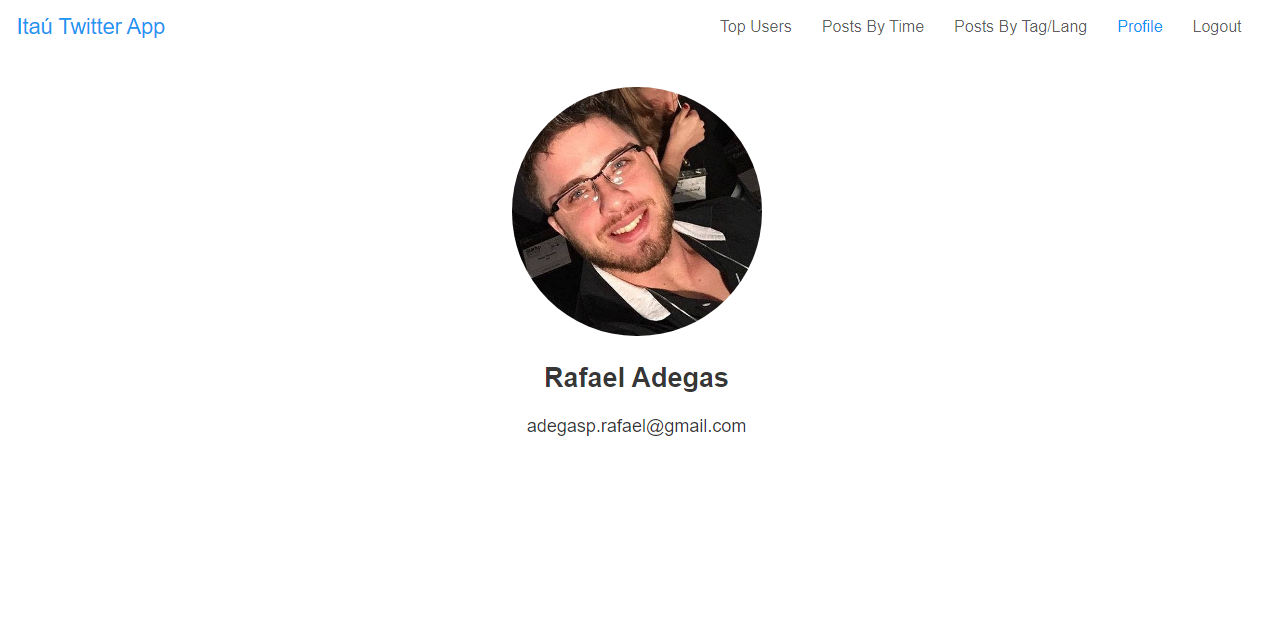
Modelo utilizando base de dados NoSql (MongoDB).



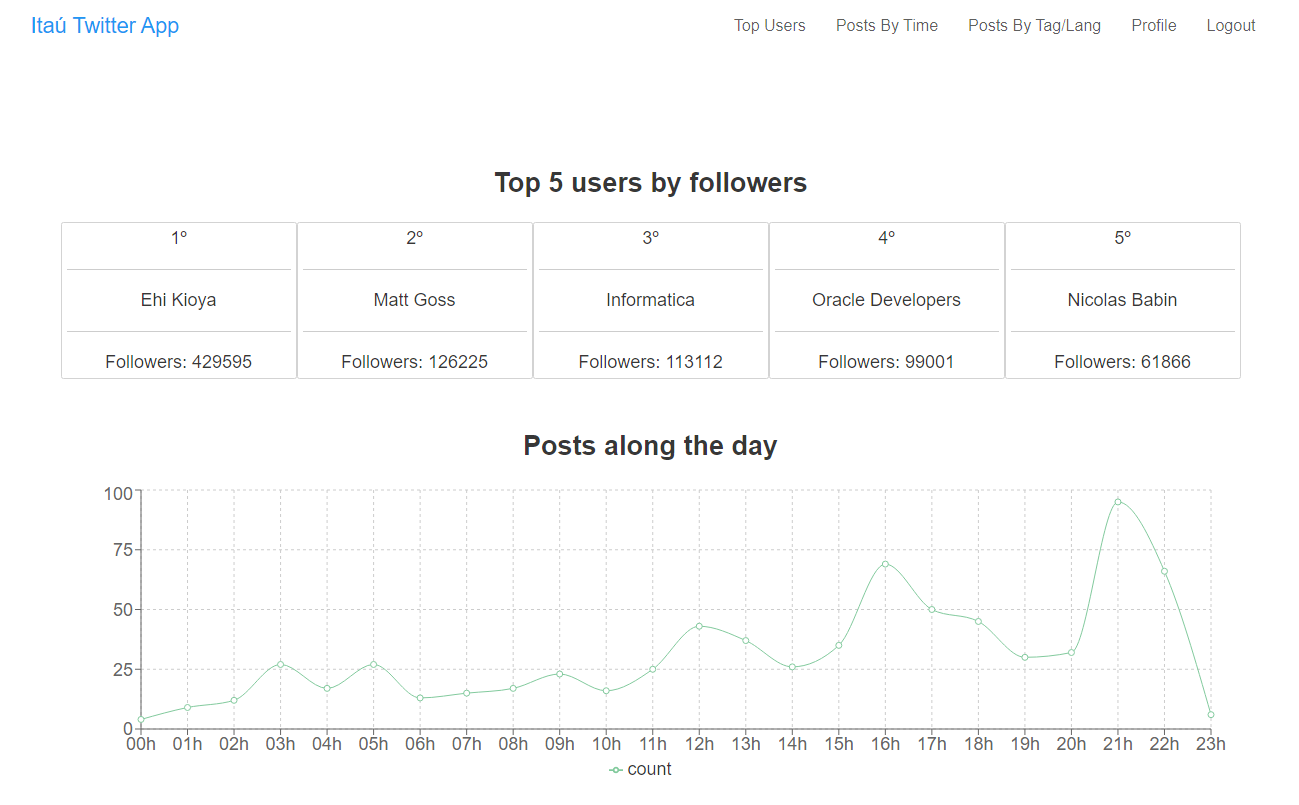
**Telas - Login**



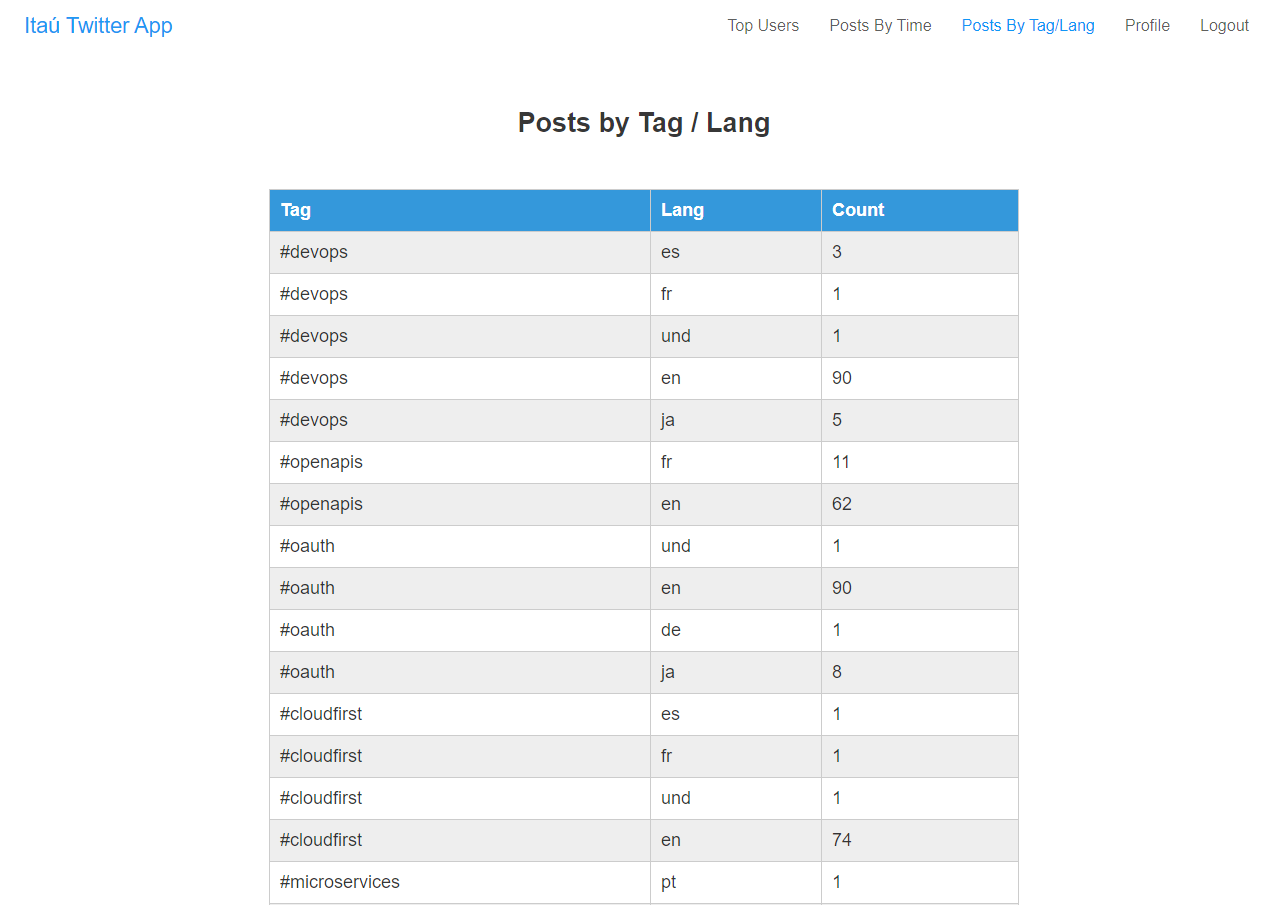
**Telas - Profile**



**Telas – Home(Top 5 Users and Posts along the day)**



**Telas – Posts By Tag and Lang**



**Referências**

<https://docs.docker.com/compose/gettingstarted/> - Docker compose reference

<https://www.digitalocean.com/community/tutorials/containerizing-a-node-js-application-for-development-with-docker-compose> - Docker compose

<https://oauth.net/2/> - Oauth2

<https://www.baeldung.com/spring-boot-mongodb-auto-generated-field> - Spring Mongodb

<http://recharts.org/en-US/api> - Graficos

[https://app.creately.com](https://app.creately.com/) – modelagem NoSql

**Contas**

GitHub: <https://github.com/adegas123>