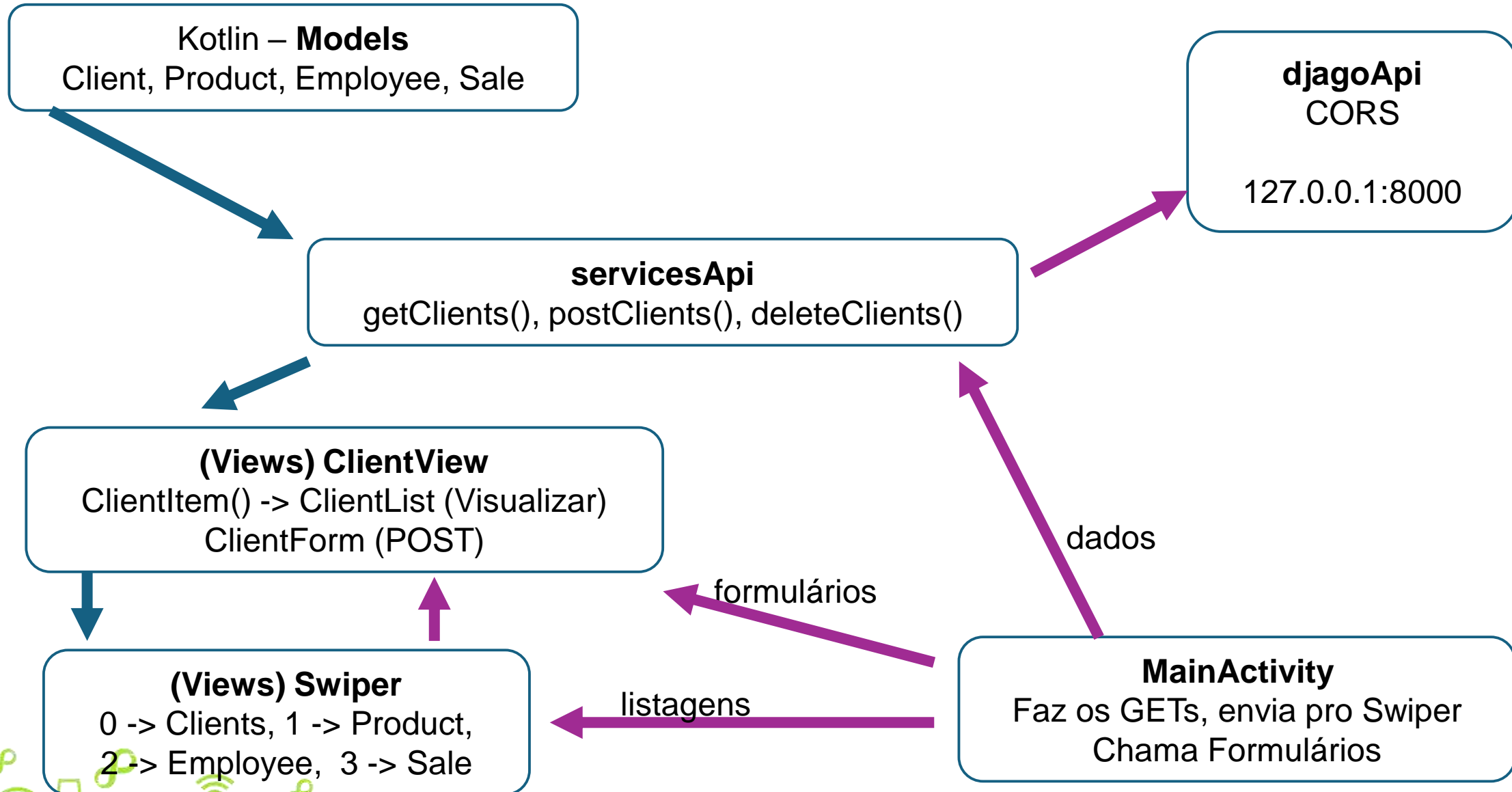


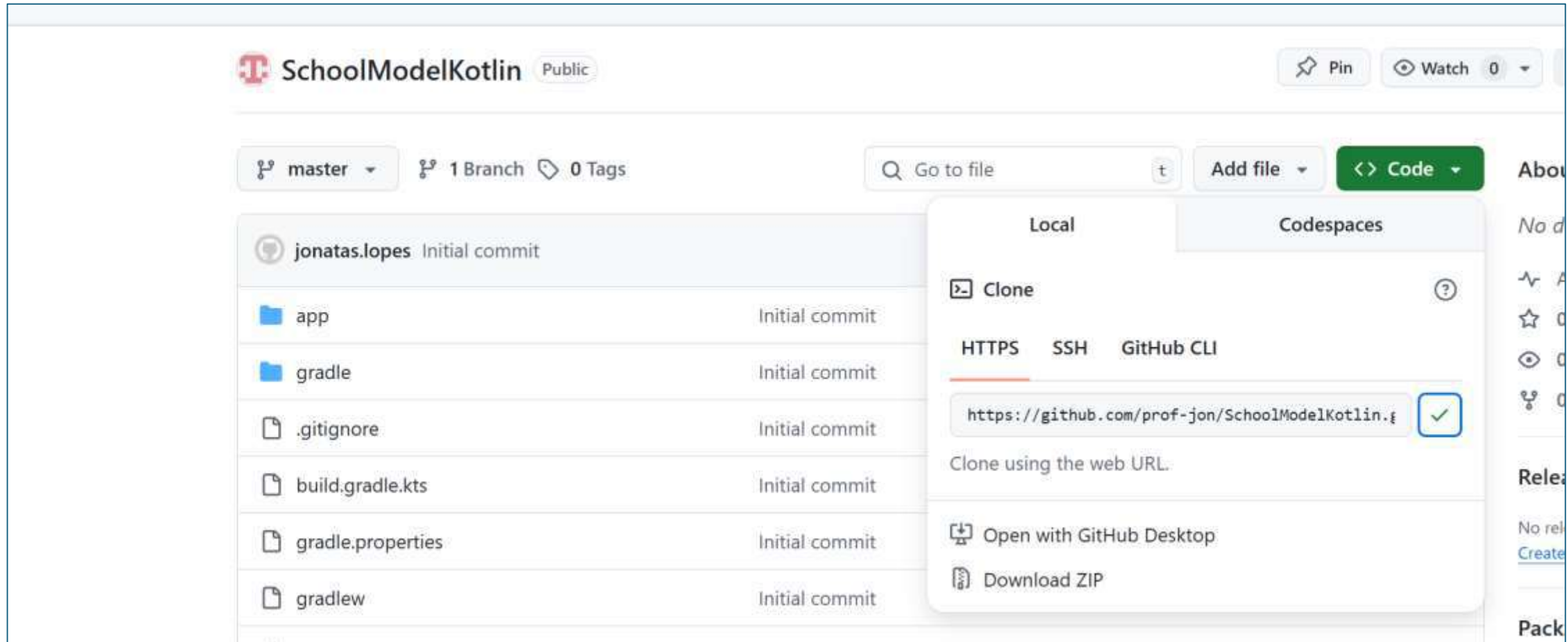


Integração com API – Consumo de Dados



Kotlin importar o Projeto

1. <https://github.com/prof-jon/SchoolModelKotlin.git>



2. Atualizar Django - <https://github.com/prof-jon/SchoolModelAPI.git>

3. RODAR, se somente views de Client - OK

Kotlin importar o Projeto

3. Descomentar: (MODELS)

apiClasses.kt x

```
1 package com.example.test.models
2
3 //import kotlinx.datetime.Instant
4 import kotlinx.serialization.Serializable
5 //import kotlinx.serialization.SerialName
6
7 @OptIn(kotlinx.serialization.InternalSerializationApi::class)
8 @Serializable
9 data class Client(
10     val id: Int,
11     val name: String,
12     val age: Int,
13     val rg: String,
14     val cpf: String
15 )
16
17 //@OptIn(kotlinx.serialization.InternalSerializationApi::class)
18 //@Serializable
19 //data class Product(
20 //    val id: Int,
```

4. SERVICE

```
// return this.http.get("http://10.0.2.2:8000/clients/").body<List>()
// }
// suspend fun getProducts(): List<Product> {
//     return this.http.get("http://10.0.2.2:8000/products/").body<List>()
// }
// suspend fun getEmployees(): List<Employee> {
//     return this.http.get("http://10.0.2.2:8000/employees/").body<List>()
// }
// suspend fun getSales(): List<SaleGet> {
//     return this.http.get("http://10.0.2.2:8000/sales/").body<List>()
// }
```

5. (VIEWS) Product, Employee, Sales, Swiper

```
//@Composable
//fun EmployeeListItem(employee: Employee, onDelete: (Employee) -> Unit) {
//    Card(
//        modifier = Modifier
//            .fillMaxWidth()
//            .padding(horizontal = 16.dp, vertical = 4.dp),
//        elevation = CardDefaults.cardElevation(4.dp)
//    ) {
//        Row(
//            modifier = Modifier
//                .fillMaxWidth()
//                .padding(16.dp),
```



Kotlin importar o Projeto

3. Descomentar: (MODELS)

apiClasses.kt x

```
1 package com.example.test.models
2
3 //import kotlinx.datetime.Instant
4 import kotlinx.serialization.Serializable
5 //import kotlinx.serialization.SerialName
6
7 @OptIn(kotlinx.serialization.InternalSerializationApi::class)
8 @Serializable
9 data class Client(
10     val id: Int,
11     val name: String,
12     val age: Int,
13     val rg: String,
14     val cpf: String
15 )
16
17 //@OptIn(kotlinx.serialization.InternalSerializationApi::class)
18 //@Serializable
19 //data class Product(
20 //    val id: Int,
```

4. SERVICE

```
// return this.http.get("http://10.0.2.2:8000/clients/").body<List>()
// }
// suspend fun getProducts(): List<Product> {
//     return this.http.get("http://10.0.2.2:8000/products/").body<List>()
// }
// suspend fun getEmployees(): List<Employee> {
//     return this.http.get("http://10.0.2.2:8000/employees/").body<List>()
// }
// suspend fun getSales(): List<SaleGet> {
//     return this.http.get("http://10.0.2.2:8000/sales/").body<List>()
// }
```

5. (VIEWS) Product, Employee, Sales, Swiper

```
//@Composable
//fun EmployeeListItem(employee: Employee, onDelete: (Employee) -> Unit) {
//    Card(
//        modifier = Modifier
//            .fillMaxWidth()
//            .padding(horizontal = 16.dp, vertical = 4.dp),
//        elevation = CardDefaults.cardElevation(4.dp)
//    ) {
//        Row(
//            modifier = Modifier
```



Kotlin importar o Projeto

6. SWIPER – deleta o numero 1 atual e descomenta o 1 final

Deletar o de azul

```
Box(modifier = Modifier.fillMaxSize()) {  
    HorizontalPager(  
        state = pagerState,  
        modifier = Modifier.fillMaxSize()  
    ) { page ->  
        when (page) {  
            0 -> ClientList(clients, onDelete = onDeleteClient)  
            //vamos ter que deletar esse numero 1 depois  
            1 -> ClientList(clients, onDelete = onDeleteClient)  
            // 1 -> ProductList(products, onDelete = onDeleteProduct)  
            // 2 -> EmployeeList(employees, onDelete = onDeleteEmployee)  
            // 3 -> SaleList(sales, onDelete = onDeleteSale)  
        }  
    }  
}
```

Kotlin importar o Projeto

6. MainActivity
descomentar tudo
Deletar o de azul

```
fun Greeting(name: String, modifier: Modifier = Modifier) {  
    currentPage?.let { page ->  
        Dialog(onDismissRequest = { currentPage = null }) {  
            ) {  
                0 -> AddClientForm { newClient ->  
                    }  
                //vamos ter que deletar esse numero 1 depois  
                1 -> AddClientForm { newClient ->  
                    coroutineScope.launch {  
                        ApiService.postClient(newClient)  
                        clients = ApiService.getClients()  
                        currentPage = null  
                    }  
                }  
            }  
        }  
    }  
    //  
    //  
    //  
    //  
    //  
    //  
    1 -> AddProductForm { newProduct ->  
        coroutineScope.launch {  
            ApiService.postProduct(newProduct)  
            products = ApiService.getProducts()  
            currentPage = null  
        }  
    }  
}
```

RODAR



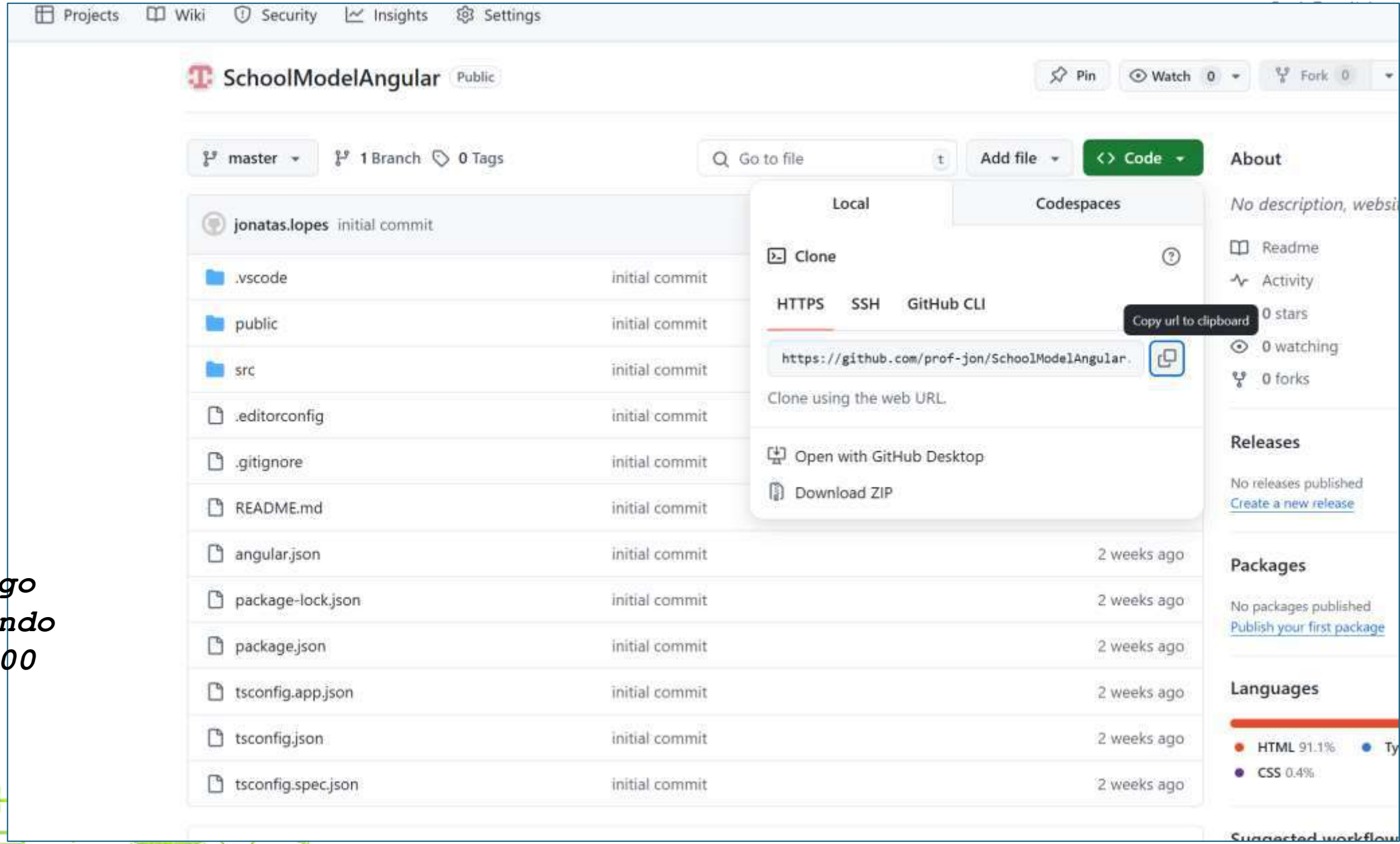
Angular importar o Projeto

1. <https://github.com/prof-jon/SchoolModelAngular.git>

2. Npm install

3. RODAR!
*se somente
views de Client
- OK*

*Lembre-te: Django
deve estar rodando
no localhost:8000*



Projects Wiki Security Insights Settings

SchoolModelAngular Public

Pin Watch 0 Fork 0

master 1 Branch 0 Tags

Go to file Add file Code

jonatas.lopes initial commit

.vscode initial commit

public initial commit

src initial commit

.editorconfig initial commit

.gitignore initial commit

README.md initial commit

angular.json initial commit 2 weeks ago

package-lock.json initial commit 2 weeks ago

package.json initial commit 2 weeks ago

tsconfig.app.json initial commit 2 weeks ago

tsconfig.json initial commit 2 weeks ago

tsconfig.spec.json initial commit 2 weeks ago

Local Codespaces

Clone

HTTPS SSH GitHub CLI

Copy url to clipboard

https://github.com/prof-jon/SchoolModelAngular.git

Clone using the web URL

Open with GitHub Desktop

Download ZIP

About

No description, website

Readme

Activity

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

HTML 91.1% TypeScript 8.9%

CSS 0.4%

Suggested workflow



Angular importar o Projeto

4.Descomentar INTERFACE

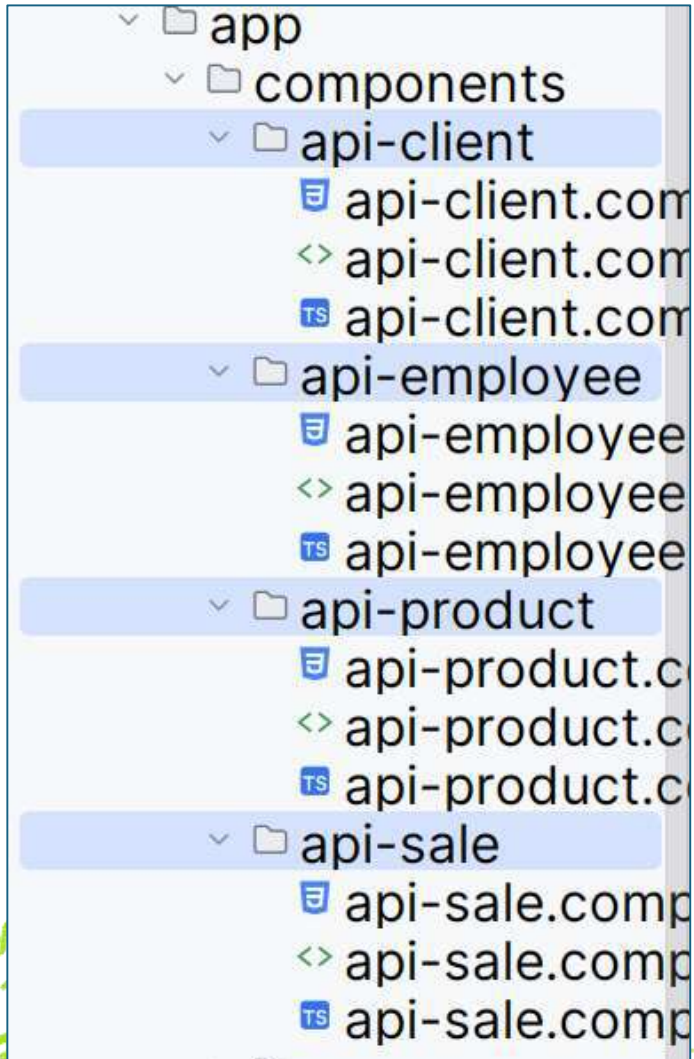
```
1 export interface Client { Show usages & jonatas.lopes
2     id: number;
3     name: string;
4     age: number;
5     rg: string;
6     cpf: string;
7 }
8
9 export interface Product { no usages & jonatas.lopes
10     id: number;
11     description: string;
12     quantity: number;
13 }
14 jonatas.lopes, Moments ago • Adicionados componentes
15 // export interface Employee {
16 //     id: number;
17 //     name: string;
18 //     registration: string;
19 //     active: boolean;
20 //     created_at: Date;
21 //     modified_at: Date;
22 // }
23 //
24 // export interface SaleGet {
25 //     id: number;
```

5.Descomentar SERVIÇO deletar Azul

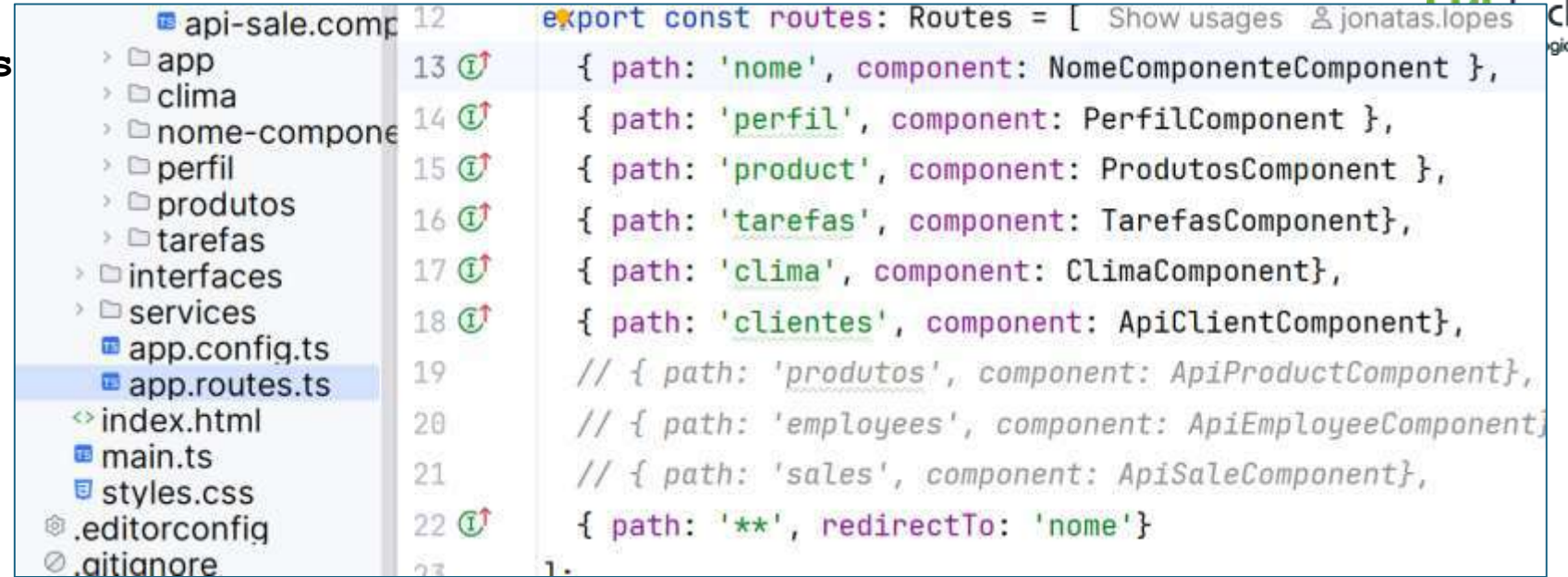
```
2 import {HttpClient} from '@angular/common/http';
3 import {Observable} from 'rxjs';
4 import {Client} from '../interfaces/Escola'; jonatas.lopes
5 // import {Client, Employee, Product, SaleGet, SalePost} fr
6
7 @Injectable({ Show usages & jonatas.lopes
8     providedIn: 'root'
9 })
10 export class EscolaService {
11     private http: HttpClient = inject(HttpClient);
12     urlbase: string = 'http://localhost:8000';
13
14     getClients(): Observable<Client[]> { Show usages & jonatas.lo
15         return this.http.get<Client[]>({ url: `${this.urlbase}/cli
16     }
17
18     // getProducts(): Observable<Product[]> {
19     //     return this.http.get<Product[]>(`${this.urlbase}/pro
20     // }
21
22     // getEmployees(): Observable<Employee[]> {
23     //     return this.http.get<Employee[]>(`${this.urlbase}/em
24     // }
25
26     // getSales(): Observable<SaleGet[]> {
27     //     return this.http.get<SaleGet[]>(`${this.urlbase}/sal
28     // }
29
30     // saveClient(client: Client) {
31     //     return this.http.post(`${this.urlbase}/clients/`, cl
```


Angular importar o Projeto

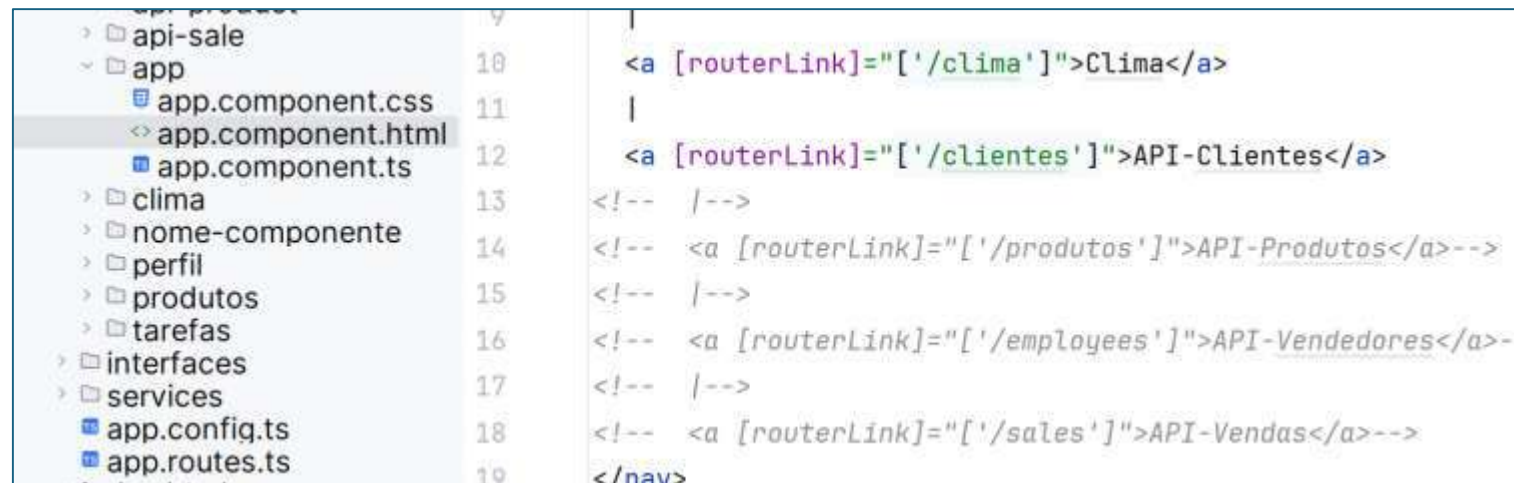
6.Descomentar Componentes HTML, CSS, .TS



7.Descomentar ROUTES



8.Descomentar Componente APP -> HTML 9.RODAR, test



Ideias de Projetos



Sugestão	Iniciante	Intermediário (validação campos)	Avançado (campos, integrações)
To-Do List (Lista de Afazeres)	CRUD básico de tarefas.	Categorias, prioridades	Deadlines e notificações
Agenda de contatos	Cadastro simples de nome, telefone e e-mail.	Busca por nome, ordenação alfabética	Upload imagem, tags, favoritos
Catálogo de Filmes	Lista com título, sinopse e imagem.	Categorias, busca e favoritos.	Consumo de API externa, Recomendação de filmes,
Blog Pessoal	CRUD de posts com título e conteúdo.	Autenticação simples, comentários.	Tags, Múltiplos autores, sistema de seguidores, (likes, dislikes)
Sistema de Votação	Criar enquetes e votar.	Visualização de resultados, login básico.	Visualização de resultados em tempo real, Restrições IP/user, agendamento de enquetes.



Ideias de Projetos



Sugestão	Iniciante	Intermediário	Avançado
Catálogo de Produtos	Lista de produtos com nome, preço	Imagem, Filtros, categorias	carrinho local, controle estoque
Sistema de Reclamações	Formulário para enviar uma reclamação.	Histórico pessoal	status da reclamação, réplicas e avaliações
Aplicativo de Estudos	Lista de tópicos baseados em matérias.	tempo de estudo, Metas de estudo (semanais, mensais)	Gráficos, lembretes, recomendações, integração API externa
Diário Financeiro	Adicionar receitas e despesas com datas.	Filtros por período, categorias, add mais campos.	Gráficos mensais e alertas.
Agenda de Eventos	Cadastro e visualização de eventos.	Inscrição, visualização no mapa.	Lembrete, ingressos esgotados, integração com calendário externo.



Obrigado!



    @fpftech.educacional