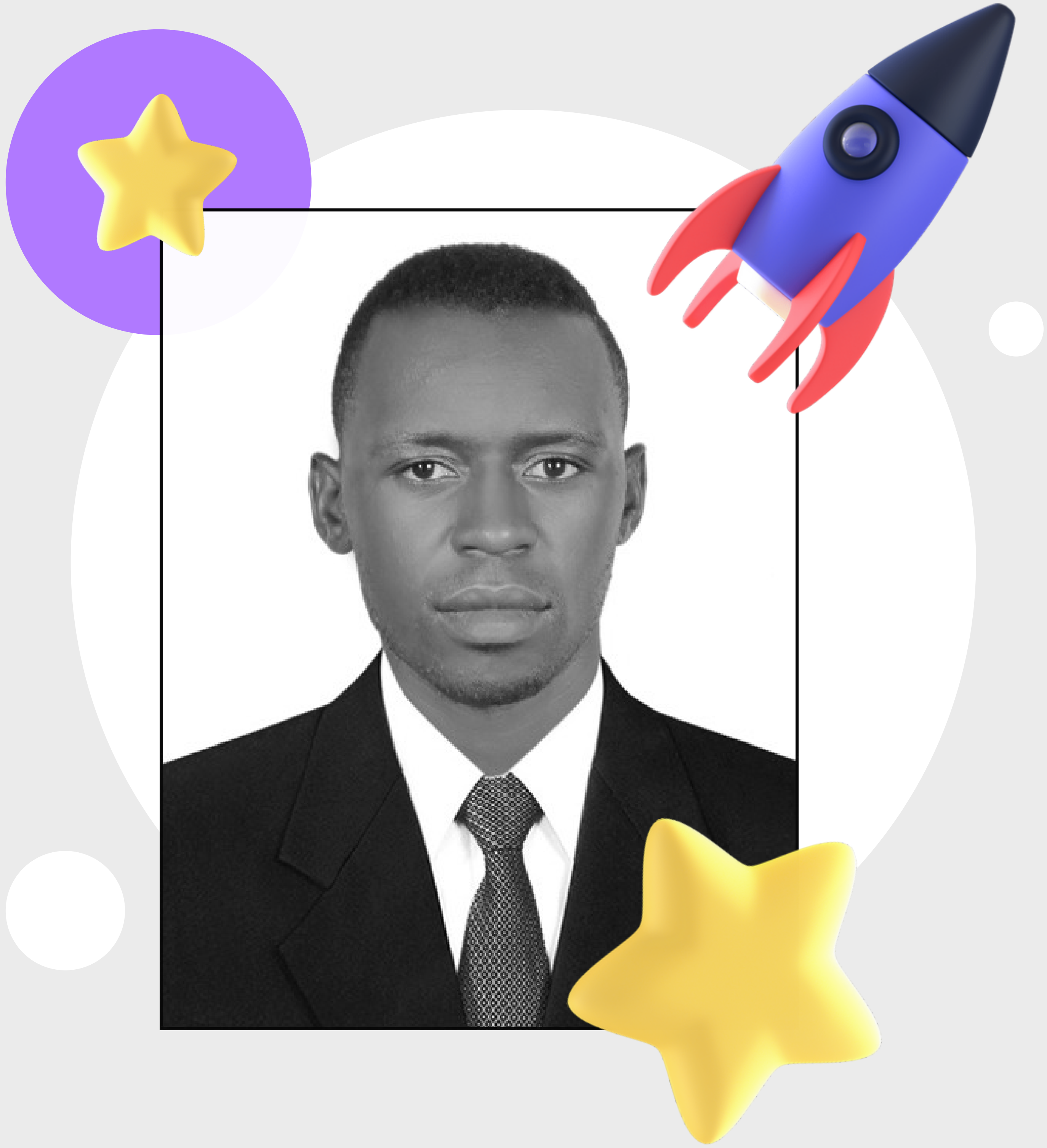
SHEY Louis CHIA



Etudiant en initial, Master-1, WEBITECH
"Développement logiciel, mobile & IoT"

Hobby : Natation, Dance, Vélo

Camerounais

https://github.com/shloch

PROJET:

DASHBOARD

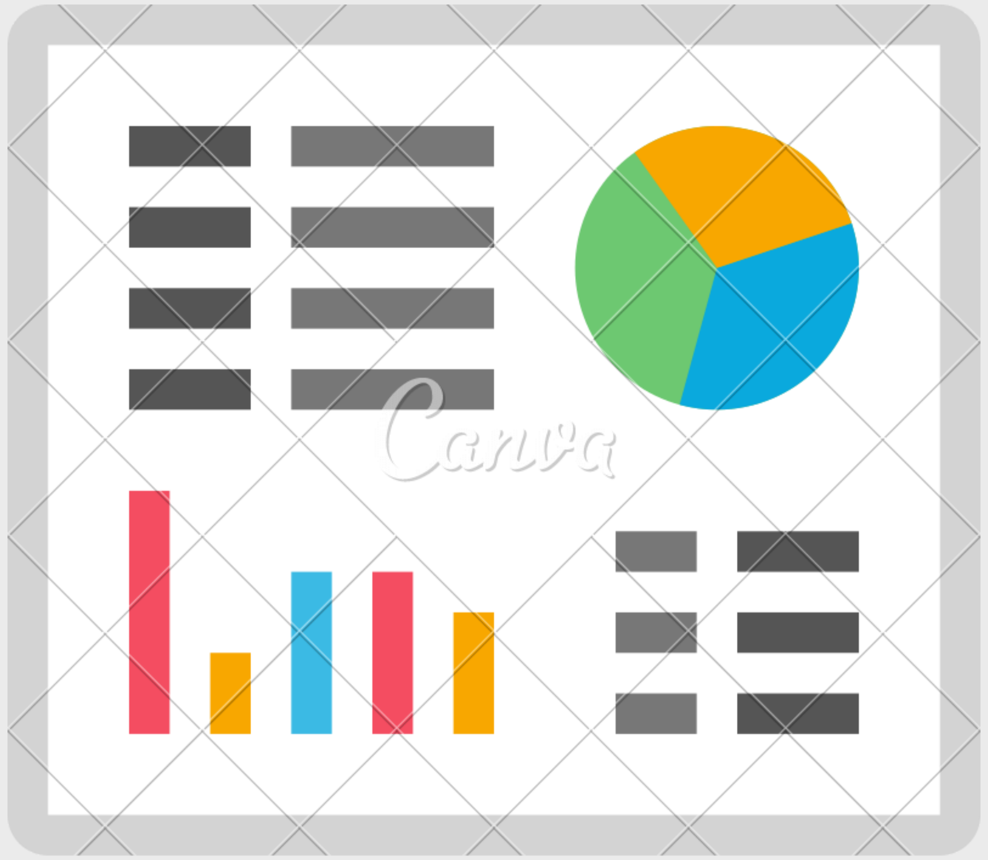
Application de mesure de performance et d'aide à la prise de décision du restaurant

« **LA PECHE** »

STACK TECHNOLOGIQUE

"LA PECHE"

DASHBOARD



BACKEND

- FrameWork : Ruby on Rails
- BDD: SQLite3
- Pas d'interface Graphique (API uniquement)

FRONT-END

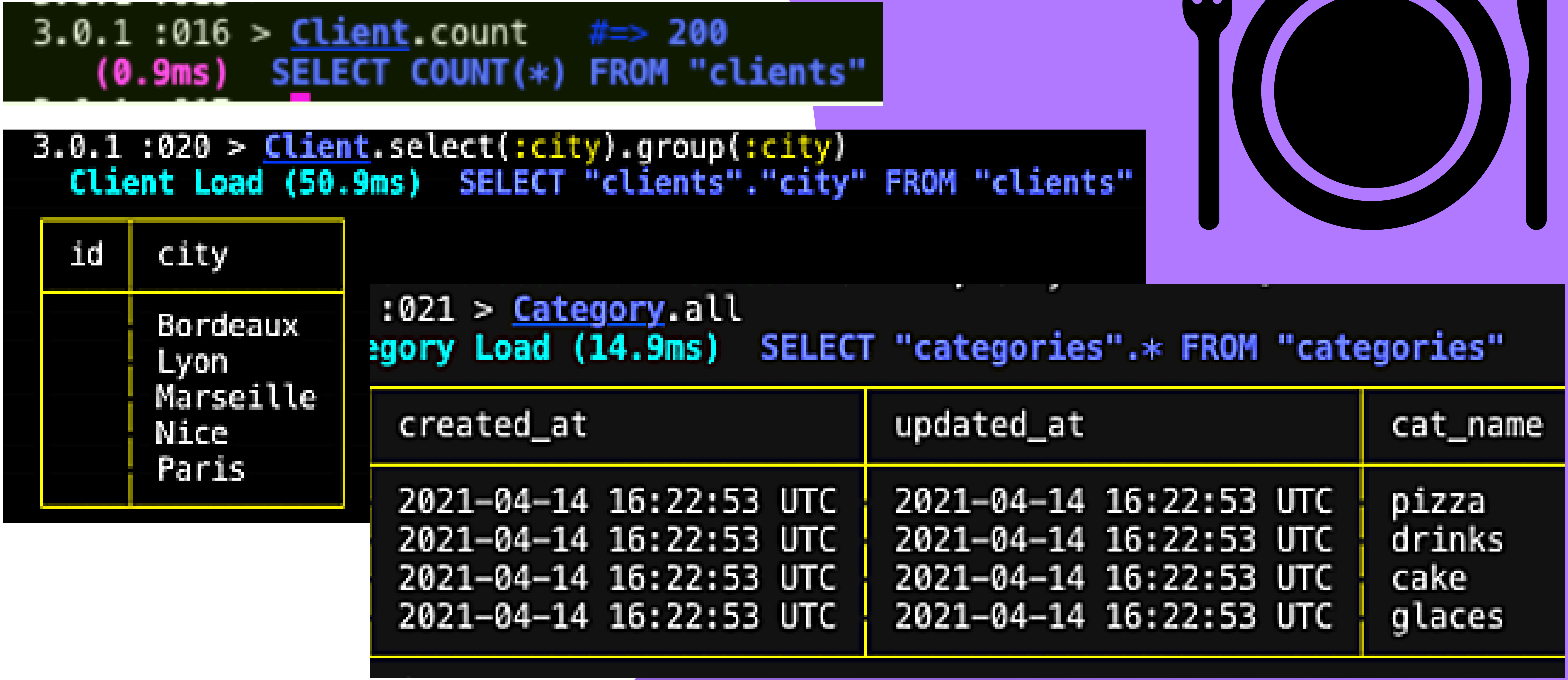
- FrameWork : ReactJs
- IHM : HTML + CSS

"LA PECHE"

https://github.com/shloch/restaurant-API-backend

"LA PECHE"

https://github.com/shloch/restaurant-API-backend



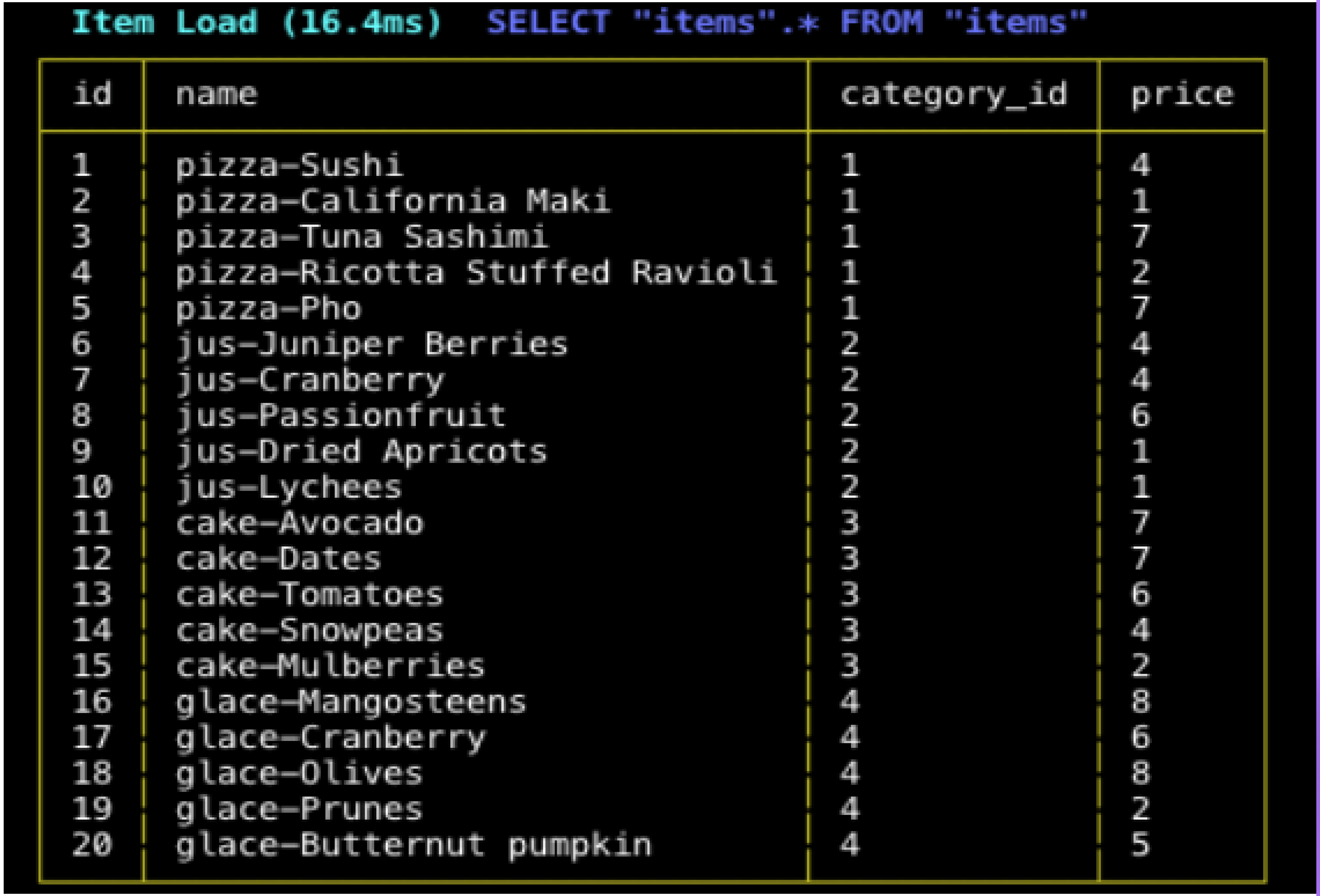
```
3.0.1 :016 > Client.count   #=> 200
   (0.9ms)  SELECT COUNT(*) FROM "clients"
```

```
3.0.1 :020 > Client.select(:city).group(:city)
   Client Load (50.9ms)  SELECT "clients"."city" FROM "clients"

 ┌────┬────────────┐
 │ id │ city       │
 ├────┼────────────┤
 │    │ Bordeaux   │
 │    │ Lyon       │
 │    │ Marseille  │
 │    │ Nice       │
 │    │ Paris      │
 └────┴────────────┘
```

```
:021 > Category.all
egory Load (14.9ms)  SELECT "categories".* FROM "categories"

 ┌─────────────────────────┬─────────────────────────┬──────────
 │ created_at              │ updated_at              │ cat_name
 ├─────────────────────────┼─────────────────────────┼──────────
 │ 2021-04-14 16:22:53 UTC │ 2021-04-14 16:22:53 UTC │ pizza
 │ 2021-04-14 16:22:53 UTC │ 2021-04-14 16:22:53 UTC │ drinks
 │ 2021-04-14 16:22:53 UTC │ 2021-04-14 16:22:53 UTC │ cake
 │ 2021-04-14 16:22:53 UTC │ 2021-04-14 16:22:53 UTC │ glaces
 └─────────────────────────┴─────────────────────────┴──────────
```

"LA PECHE"

https://github.com/shloch/restaurant-API-backend

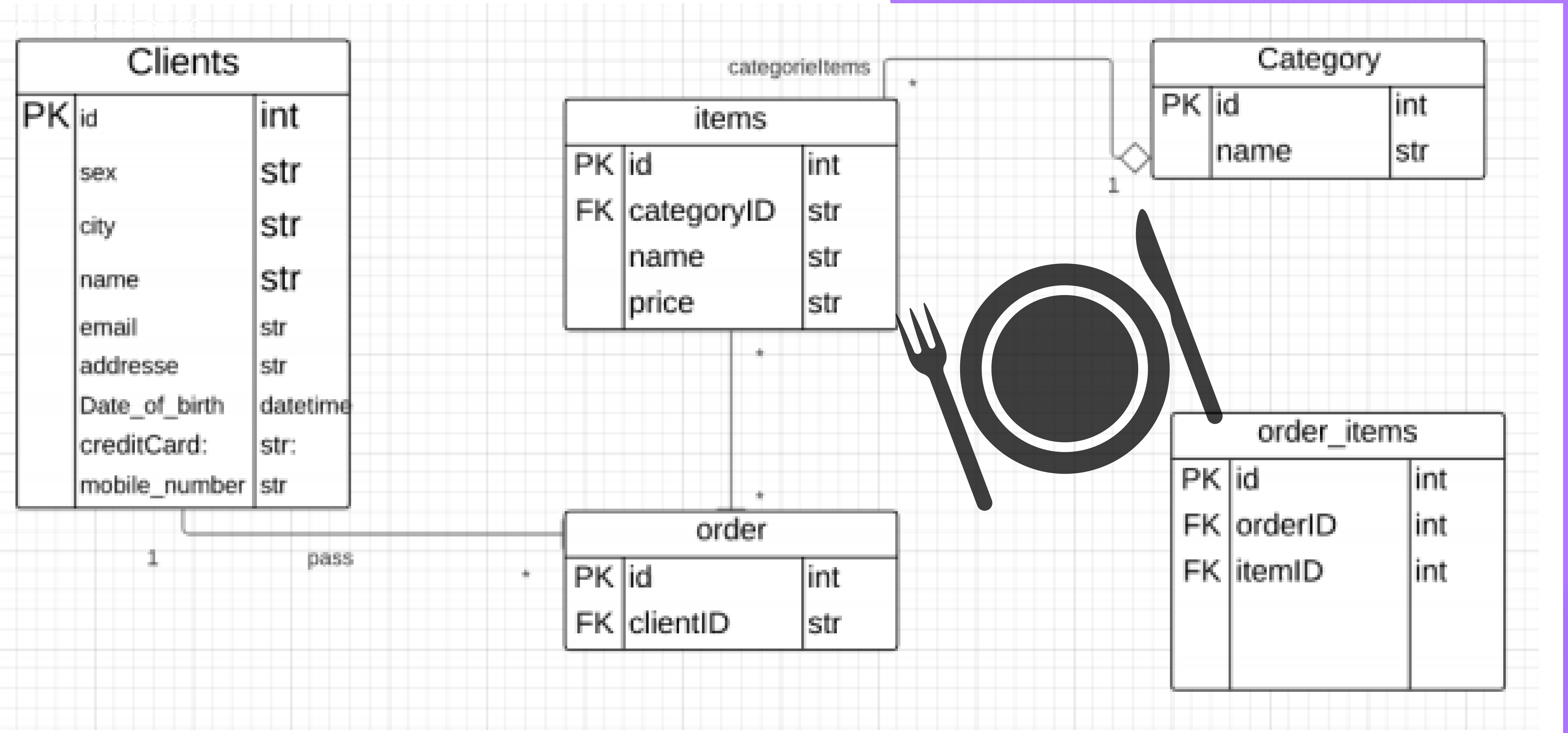


```
Item Load (16.4ms)  SELECT "items".* FROM "items"
+----+--------------------------------+-------------+-------+
| id | name                           | category_id | price |
+----+--------------------------------+-------------+-------+
| 1  | pizza-Sushi                    | 1           | 4     |
| 2  | pizza-California Maki          | 1           | 1     |
| 3  | pizza-Tuna Sashimi             | 1           | 7     |
| 4  | pizza-Ricotta Stuffed Ravioli  | 1           | 2     |
| 5  | pizza-Pho                      | 1           | 7     |
| 6  | jus-Juniper Berries            | 2           | 4     |
| 7  | jus-Cranberry                  | 2           | 4     |
| 8  | jus-Passionfruit               | 2           | 6     |
| 9  | jus-Dried Apricots             | 2           | 1     |
| 10 | jus-Lychees                    | 2           | 1     |
| 11 | cake-Avocado                   | 3           | 7     |
| 12 | cake-Dates                     | 3           | 7     |
| 13 | cake-Tomatoes                  | 3           | 6     |
| 14 | cake-Snowpeas                  | 3           | 4     |
| 15 | cake-Mulberries                | 3           | 2     |
| 16 | glace-Mangosteens              | 4           | 8     |
| 17 | glace-Cranberry                | 4           | 6     |
| 18 | glace-Olives                   | 4           | 8     |
| 19 | glace-Prunes                   | 4           | 2     |
| 20 | glace-Butternut pumpkin        | 4           | 5     |
+----+--------------------------------+-------------+-------+
```

"LA PECHE"

https://github.com/shloch/restaurant-API-backend

Diagramme de classe simplifié



Clients

PK	id	int
	sex	str
	city	str
	name	str
	email	str
	addresse	str
	Date_of_birth	datetime
	creditCard:	str:
	mobile_number	str

items

PK	id	int
FK	categoryID	str
	name	str
	price	str

Category

PK	id	int
	name	str

order

PK	id	int
FK	clientID	str

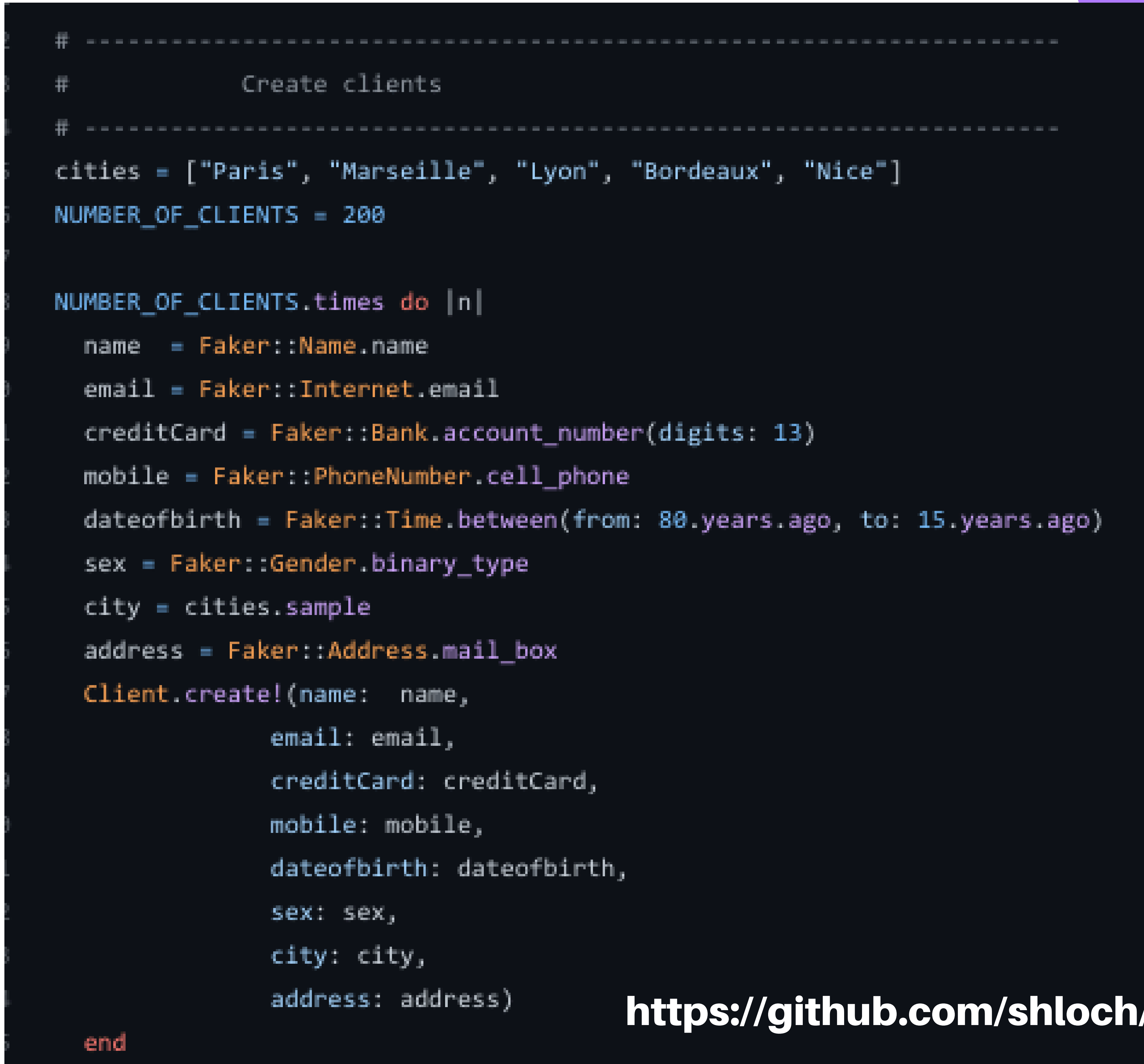
order_items

PK	id	int
FK	orderID	int
FK	itemID	int

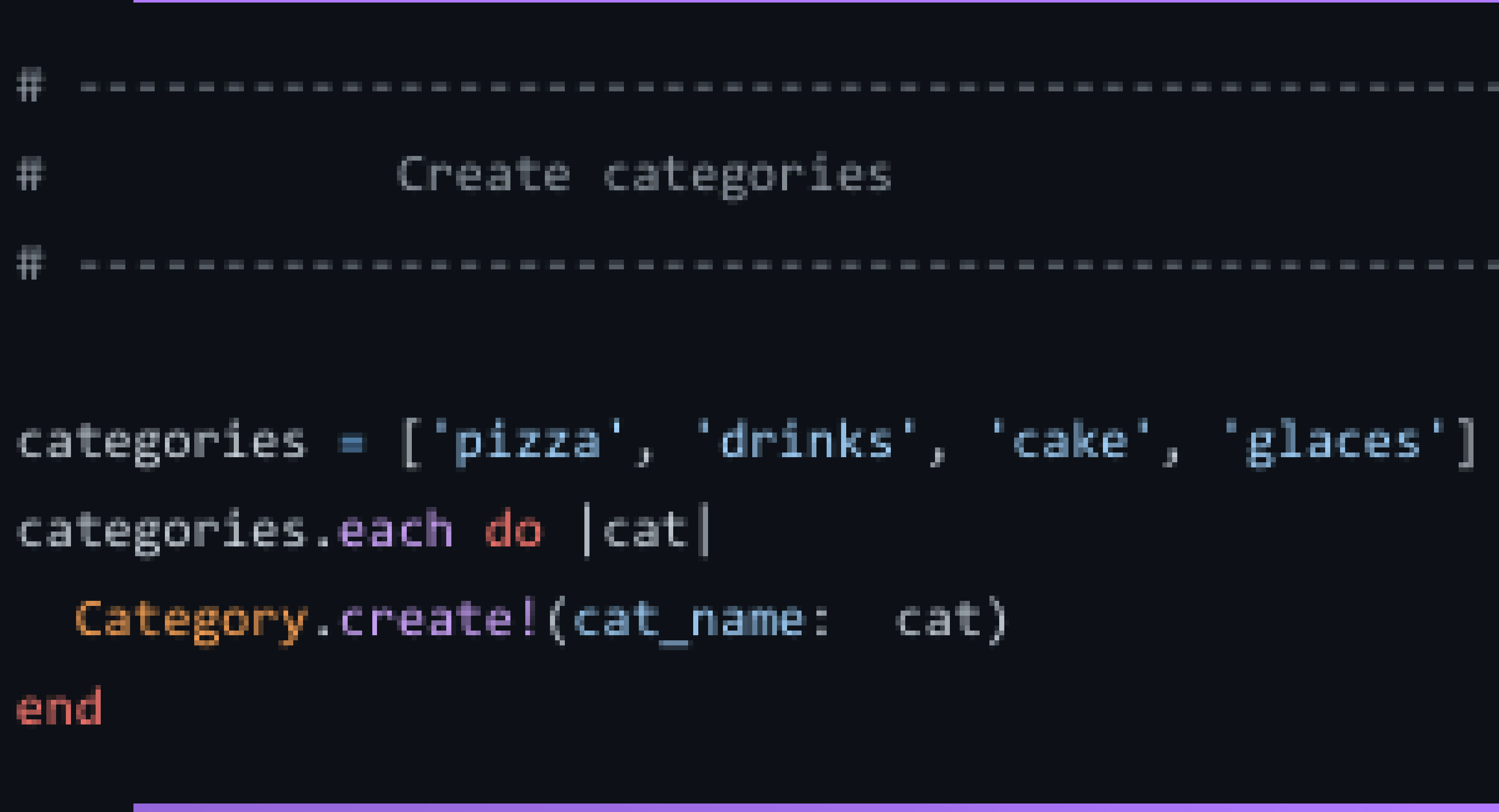
"LA PECHE"

https://github.com/shloch/restaurant-API-backend

Chargement des données dans la base de données



```ruby
# ---------------------------------------------------------------------
#             Create clients
# ---------------------------------------------------------------------
cities = ["Paris", "Marseille", "Lyon", "Bordeaux", "Nice"]
NUMBER_OF_CLIENTS = 200

NUMBER_OF_CLIENTS.times do |n|
  name   = Faker::Name.name
  email = Faker::Internet.email
  creditCard = Faker::Bank.account_number(digits: 13)
  mobile = Faker::PhoneNumber.cell_phone
  dateofbirth = Faker::Time.between(from: 80.years.ago, to: 15.years.ago)
  sex = Faker::Gender.binary_type
  city = cities.sample
  address = Faker::Address.mail_box
  Client.create!(name:  name,
                 email: email,
                 creditCard: creditCard,
                 mobile: mobile,
                 dateofbirth: dateofbirth,
                 sex: sex,
                 city: city,
                 address: address)
  end
```



```ruby
# ----------------------------------------------------------
#             Create categories
# ----------------------------------------------------------

categories = ['pizza', 'drinks', 'cake', 'glaces']
categories.each do |cat|
  Category.create!(cat_name:  cat)
end
```

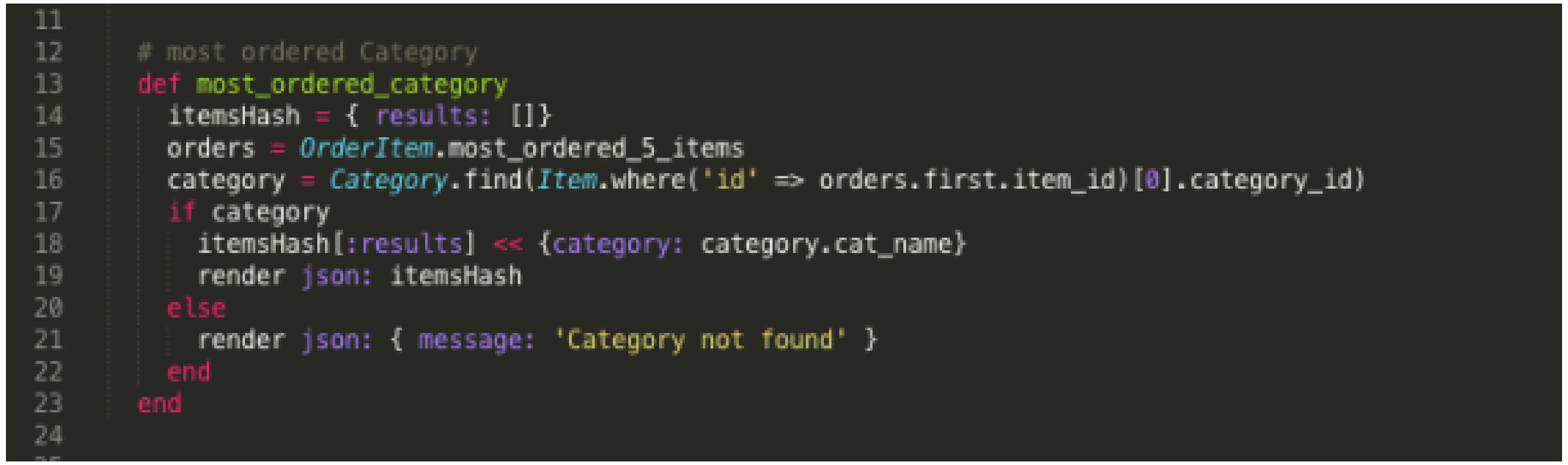
https://github.com/shloch/restaurant-API-backend/blob/master/db/seeds.rb

"LA PECHE"

https://github.com/shloch/restaurant-API-backend

Mise en place des Web services

I. Web service pour ressortir la catégorie de produit la plus commandée :

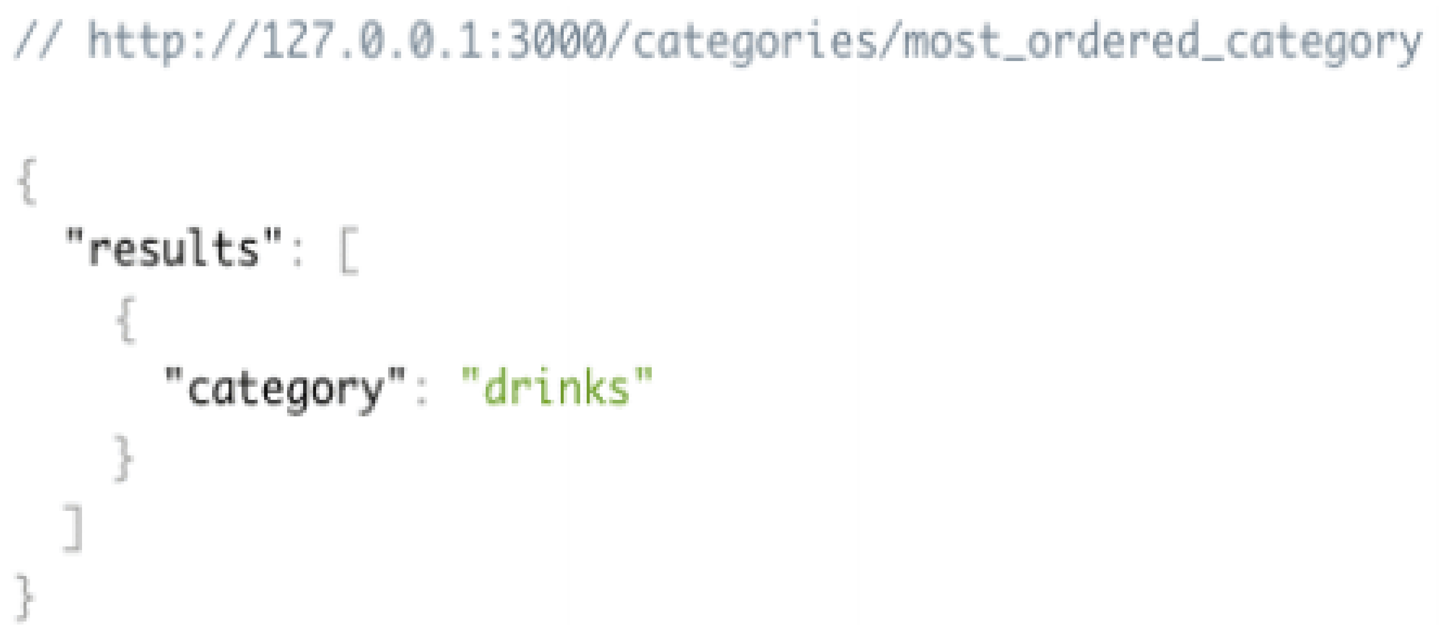


```ruby

# most ordered Category
def most_ordered_category
  itemsHash = { results: []}
  orders = OrderItem.most_ordered_5_items
  category = Category.find(Item.where('id' => orders.first.item_id)[0].category_id)
  if category
    itemsHash[:results] << {category: category.cat_name}
    render json: itemsHash
  else
    render json: { message: 'Category not found' }
  end
end

```

Accessible via l'URL

`*http://127.0.0.1:3000/categories/most_ordered_category*`.

Elle produit le resultat JSON ci-dessous:



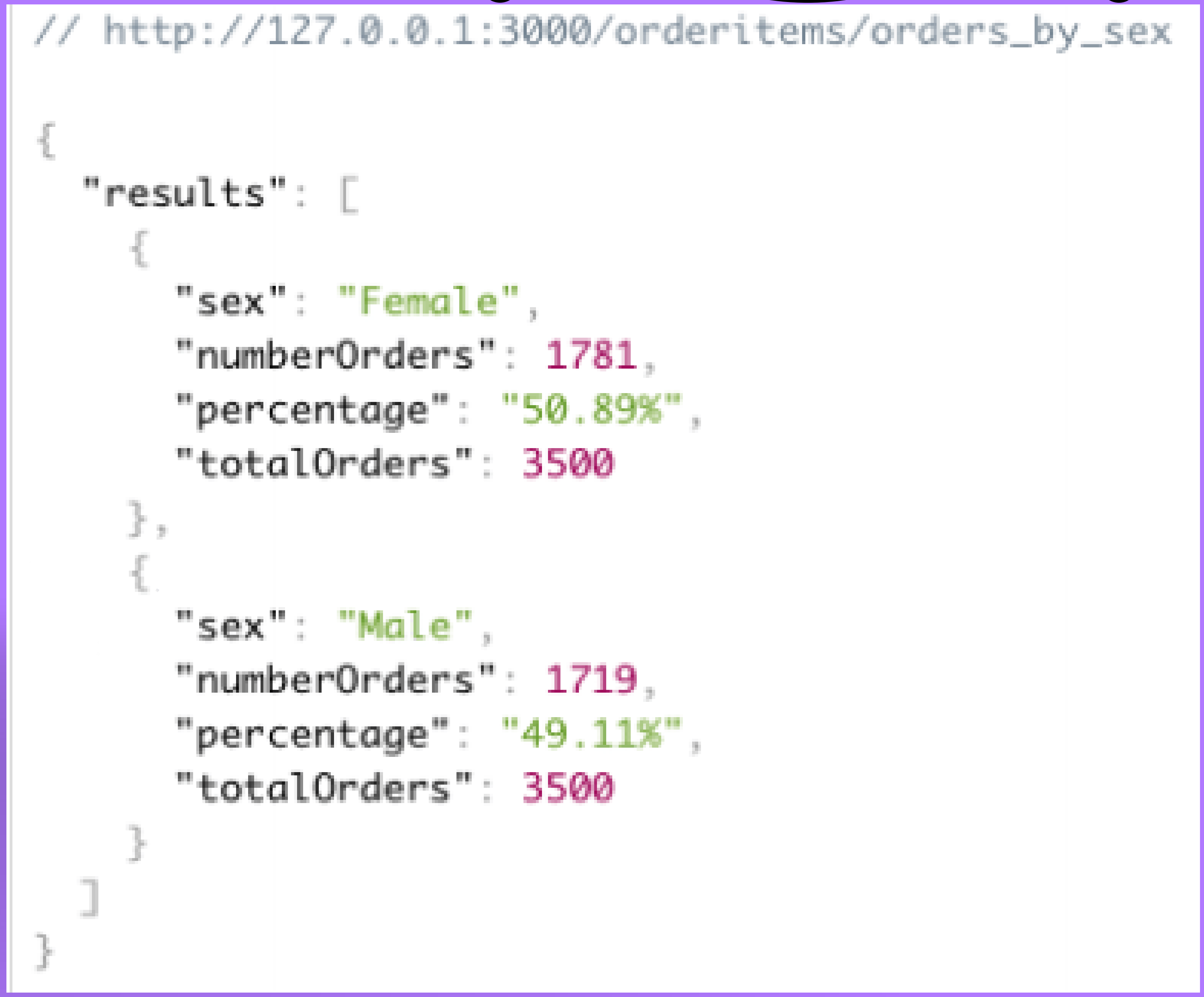
```
// http://127.0.0.1:3000/categories/most_ordered_category

{
  "results": [
    {
      "category": "drinks"
    }
  ]
}
```

"LA PECHE"

https://github.com/shloch/restaurant-API-backend

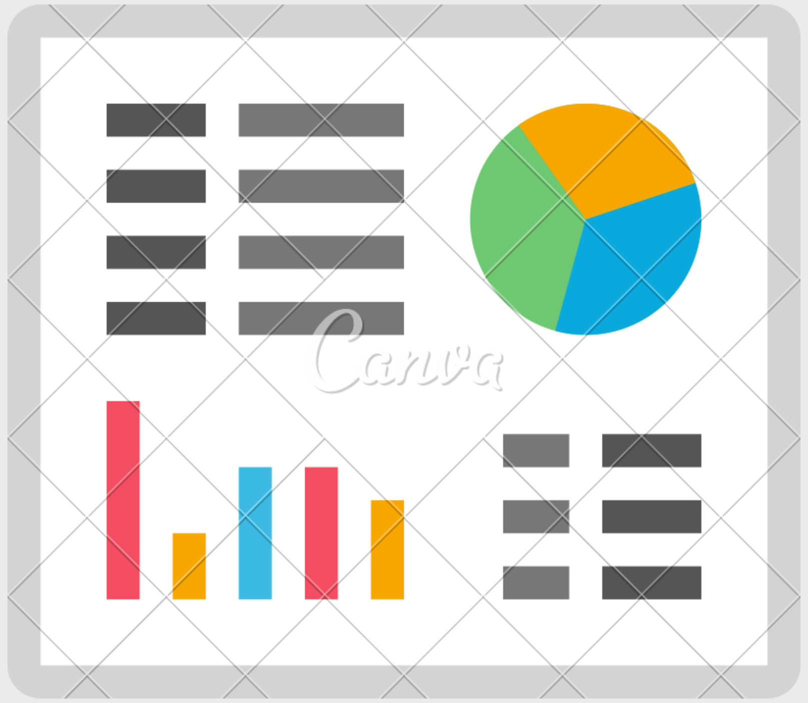
D'autres Web services

- /orderitems/highest_spender
- /categories
- /items
- /orders/total_orders
- /clients/total_clients
- /orderitems/spending_amounts_by_age_group
- /orderitems/orders_by_city
- /orderitems/orders_by_sex



```
// http://127.0.0.1:3000/orderitems/orders_by_sex

{
  "results": [
    {
      "sex": "Female",
      "numberOrders": 1781,
      "percentage": "50.89%",
      "totalOrders": 3500
    },
    {
      "sex": "Male",
      "numberOrders": 1719,
      "percentage": "49.11%",
      "totalOrders": 3500
    }
  ]
}
```

"DASHBOARD"

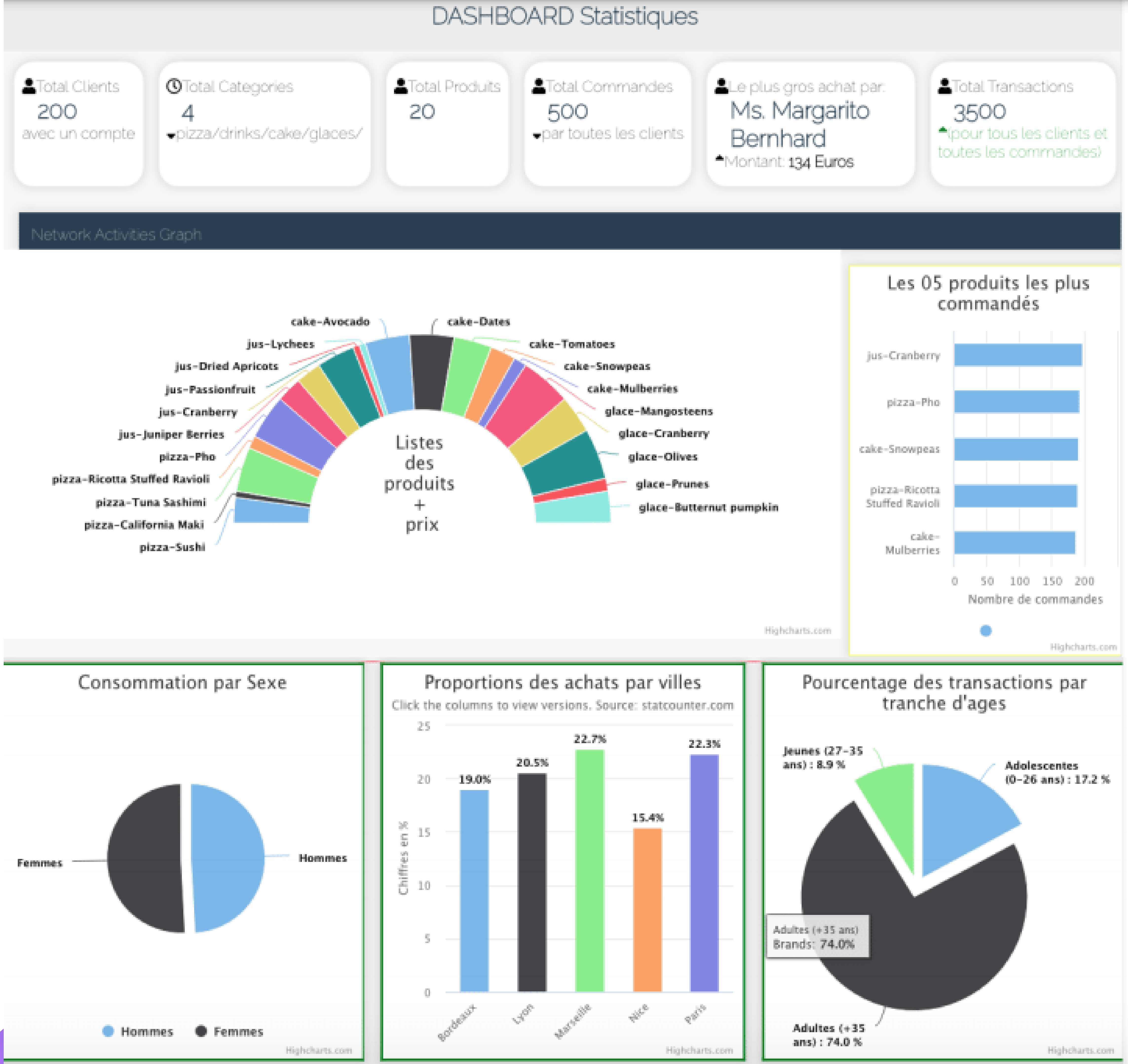


Dashboard est une application Frontend qui va permettre la consommation des web services développés dans “LA PECHE”. La technologie utilisée ici est REACTJs.

https://github.com/shloch/restaurant-DASHBOARD-FrontEnd

"DASHBOARD"

https://github.com/shloch/restaurant-DASHBOARD-FrontEnd



DASHBOARD Statistiques

Total Clients 200 avec un compte

Total Categories 4 pizza/drinks/cake/glaces/

Total Produits 20

Total Commandes 500 par toutes les clients

Le plus gros achat par: Ms. Margarito Bernhard Montant: 134 Euros

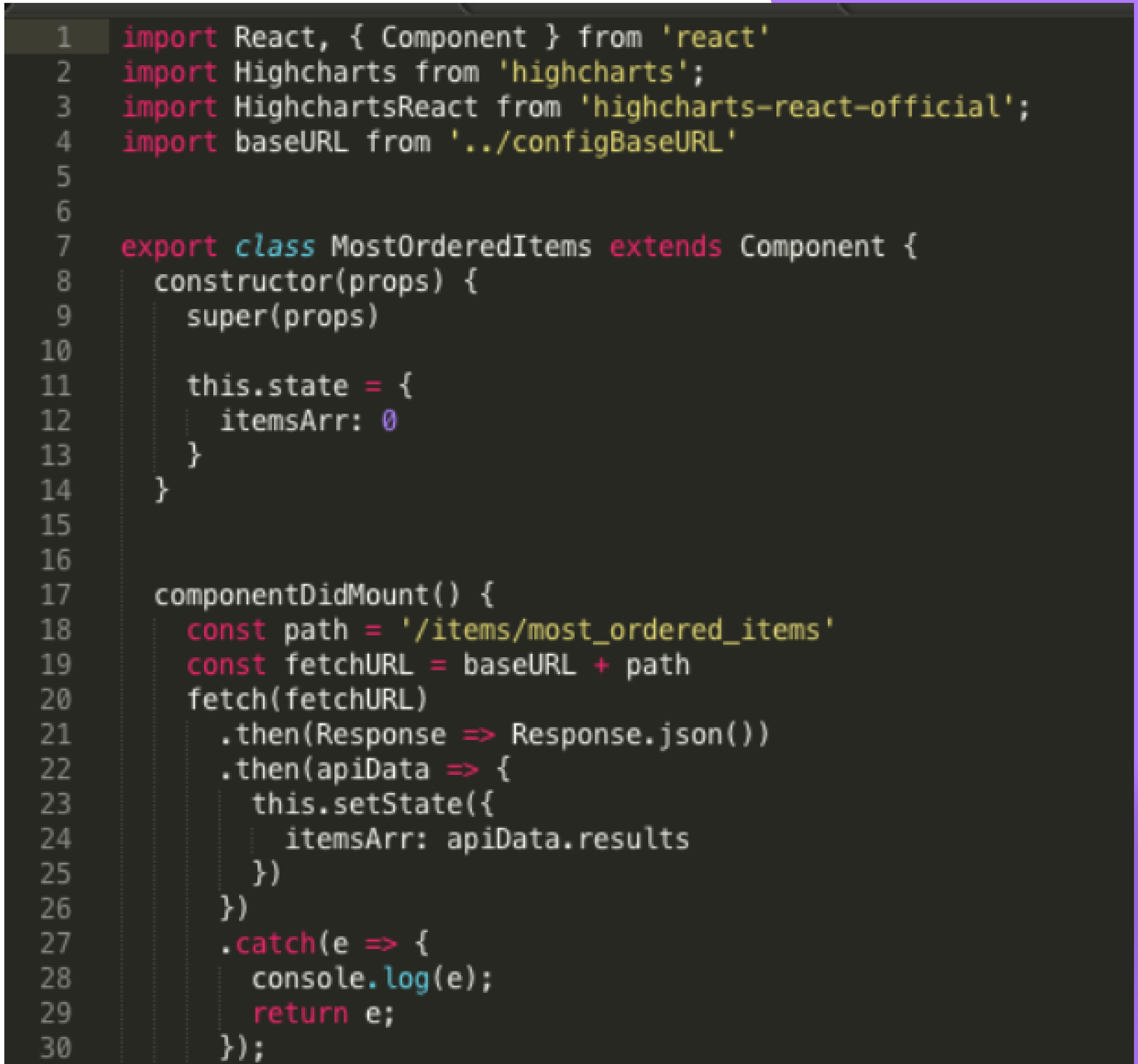
Total Transactions 3500 (pour tous les clients et toutes les commandes)

Network Activities Graph

"DASHBOARD"

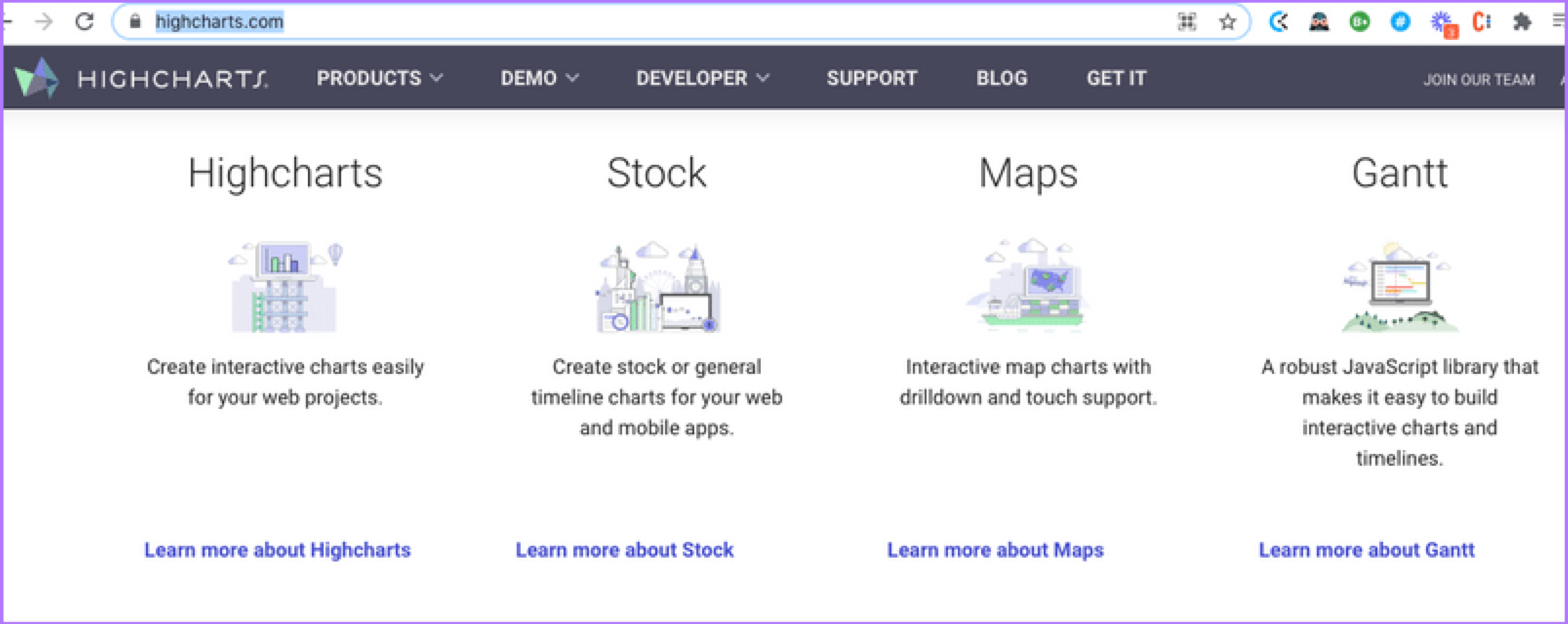
https://github.com/shloch/restaurant-DASHBOARD-FrontEnd

Presentation d'un exemple de de Composant Reactjs pour DASHBOARD



```
import React, { Component } from 'react'
import Highcharts from 'highcharts';
import HighchartsReact from 'highcharts-react-official';
import baseURL from '../configBaseURL'


export class MostOrderedItems extends Component {
  constructor(props) {
    super(props)

    this.state = {
      itemsArr: 0
    }
  }


  componentDidMount() {
    const path = '/items/most_ordered_items'
    const fetchURL = baseURL + path
    fetch(fetchURL)
      .then(Response => Response.json())
      .then(apiData => {
        this.setState({
          itemsArr: apiData.results
        })
      })
      .catch(e => {
        console.log(e);
        return e;
      });
```

LES GRAPHES AVEC "HIGHCHART"



Highcharts

Create interactive charts easily for your web projects.

Learn more about Highcharts

Stock

Create stock or general timeline charts for your web and mobile apps.

Learn more about Stock

Maps

Interactive map charts with drilldown and touch support.

Learn more about Maps

Gantt

A robust JavaScript library that makes it easy to build interactive charts and timelines.

Learn more about Gantt

LES AMELIORATIONS FUTURES

- Ecrire des testes
- Heberger "La Peche"

(Lien DASHBOARD : https://youthful-clarke-4eccc4.netlify.app/)

- Plus de securité avec **OAUTH2** : système d’authentification entre le serveur backend et le frontend
- Reflechir sur un mécanisme pour rendre DASHBOARD utile pour d'autres projets

LESSONS APPRISES

- Gestion de Projet
- Renforcement de capacité en Dévélopement
- Utilisation des nouveaux outils

Merci
WEBITECH