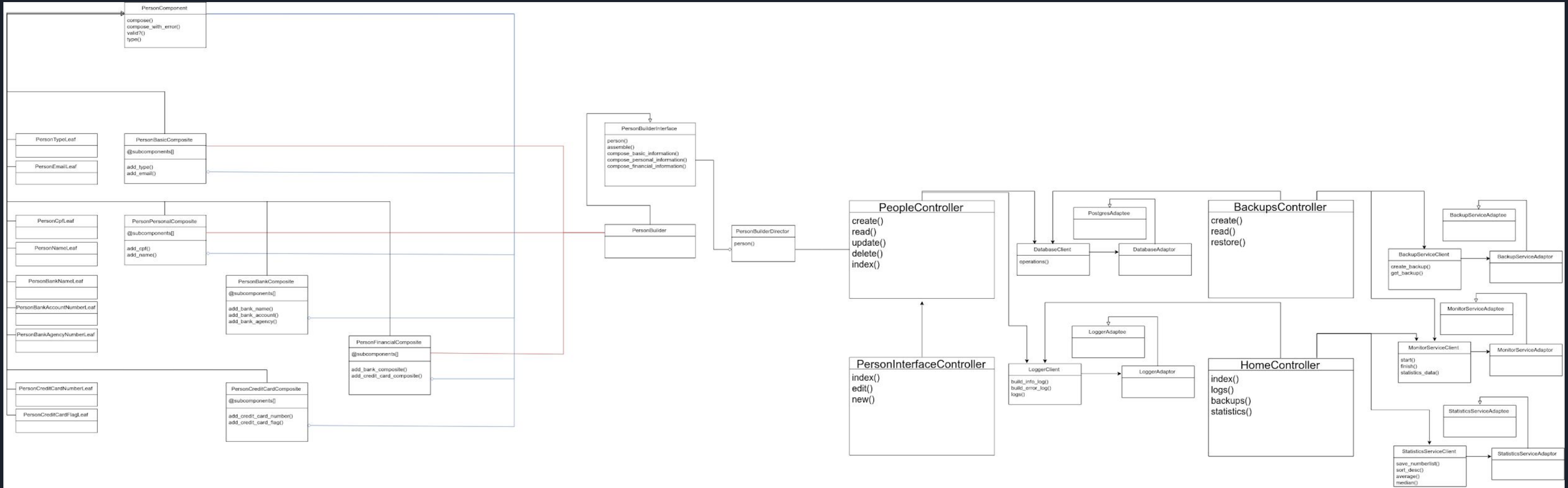
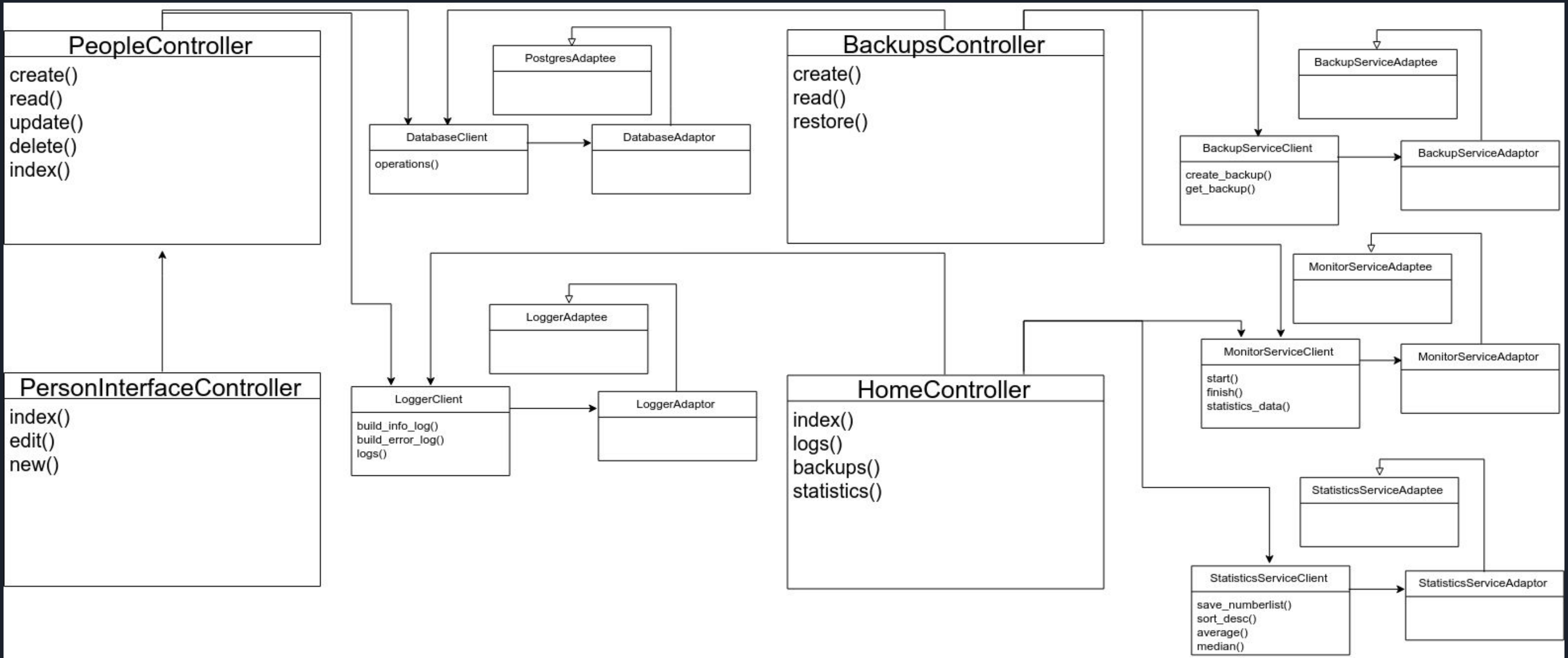
API de pessoas: integrações

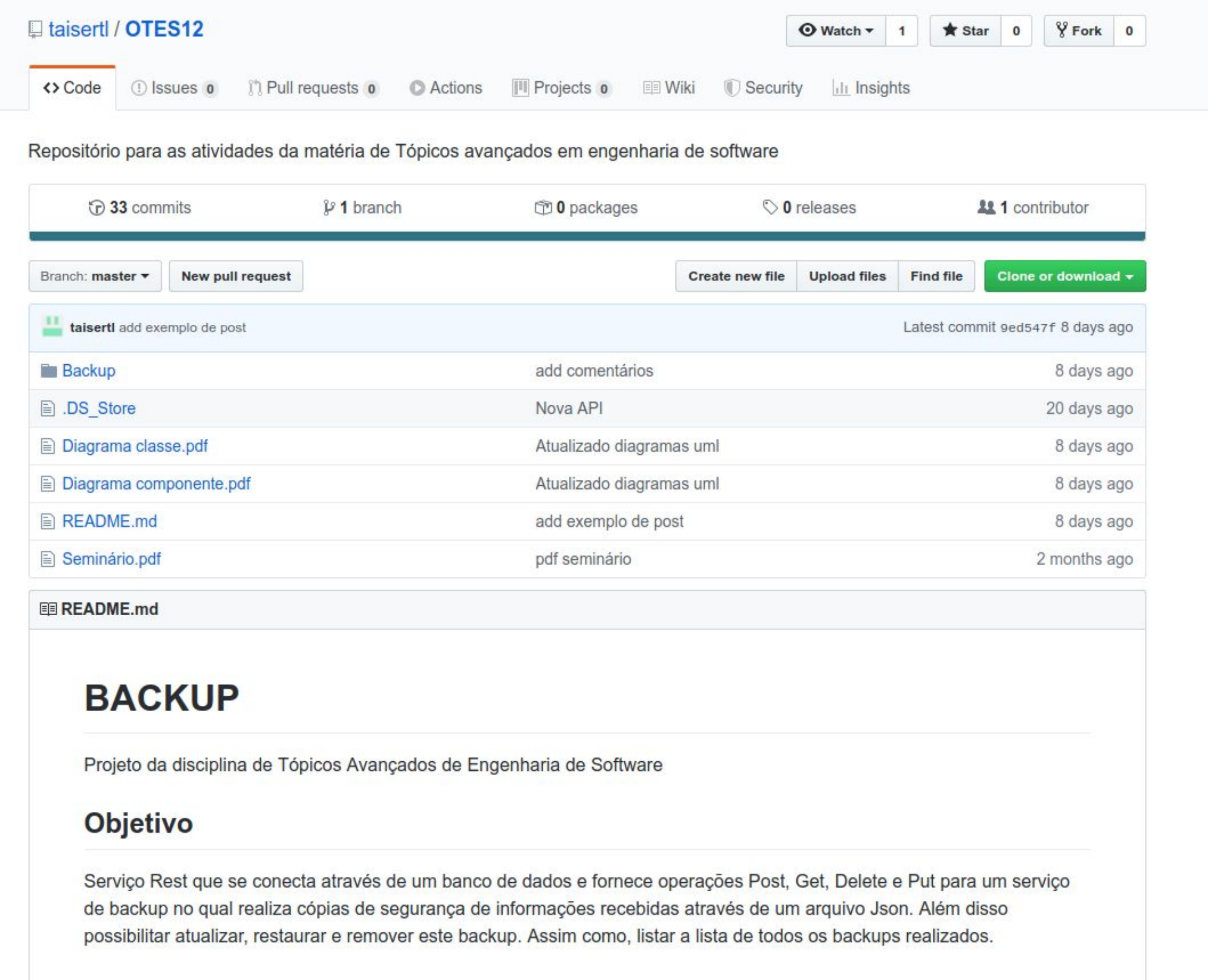
Pedro Germano Cervi

Diagrama de classes



Integrações





Repositório para as atividades da matéria de Tópicos avançados em engenharia de software

taisertl add exemplo de post — Latest commit 9ed547f 8 days ago

Backup	add comentários	8 days ago
.DS_Store	Nova API	20 days ago
Diagrama classe.pdf	Atualizado diagramas uml	8 days ago
Diagrama componente.pdf	Atualizado diagramas uml	8 days ago
README.md	add exemplo de post	8 days ago
Seminário.pdf	pdf seminário	2 months ago

README.md

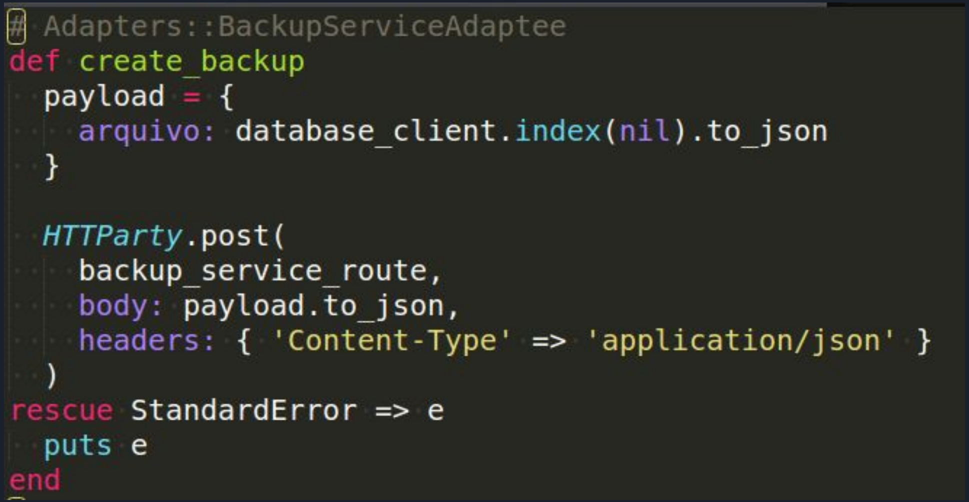
BACKUP

Projeto da disciplina de Tópicos Avançados de Engenharia de Software

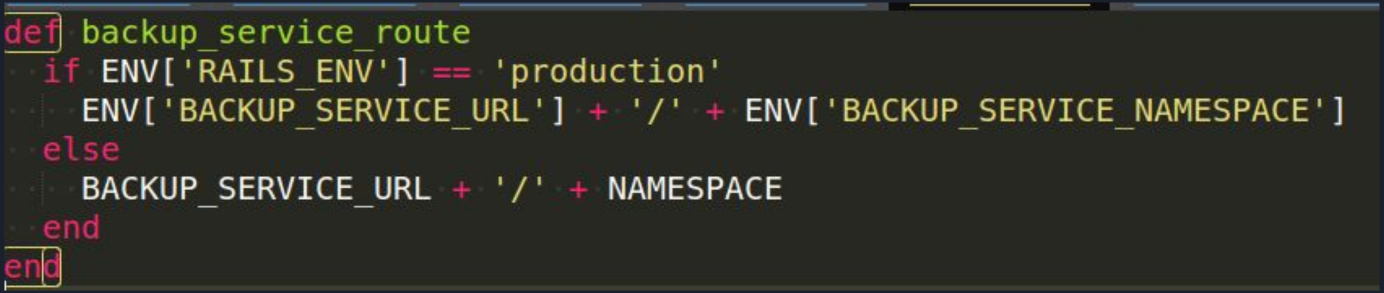
Objetivo

Serviço Rest que se conecta através de um banco de dados e fornece operações Post, Get, Delete e Put para um serviço de backup no qual realiza cópias de segurança de informações recebidas através de um arquivo Json. Além disso possibilitar atualizar, restaurar e remover este backup. Assim como, listar a lista de todos os backups realizados.

Backups - Código (chamada API)



```ruby
# Adapters::BackupServiceAdaptee
def create_backup
  payload = {
    arquivo: database_client.index(nil).to_json
  }

  HTTParty.post(
    backup_service_route,
    body: payload.to_json,
    headers: { 'Content-Type' => 'application/json' }
  )
rescue StandardError => e
  puts e
end
```

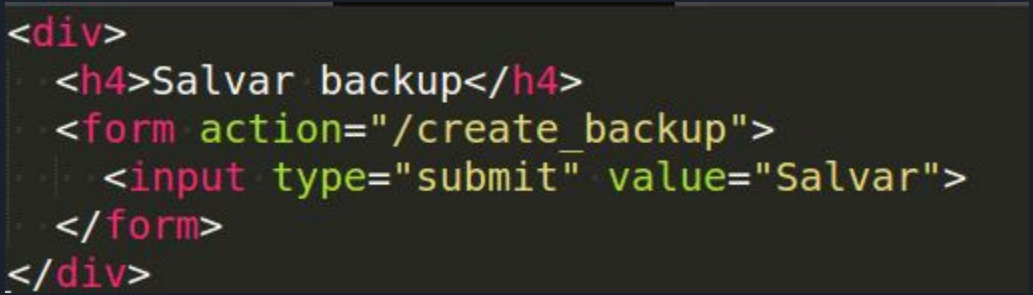


```ruby
def backup_service_route
  if ENV['RAILS_ENV'] == 'production'
    ENV['BACKUP_SERVICE_URL'] + '/' + ENV['BACKUP_SERVICE_NAMESPACE']
  else
    BACKUP_SERVICE_URL + '/' + NAMESPACE
  end
end
```

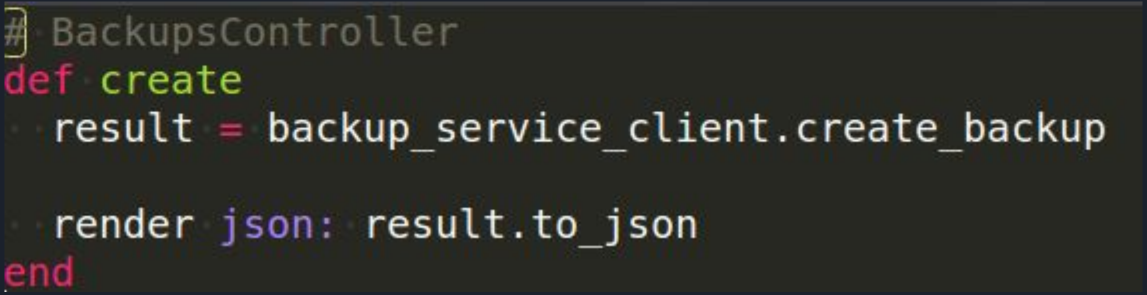
Heroku config vars

https://dashboard.heroku.com/apps/salty-cove-05656/settings

Backups - Código (uso)



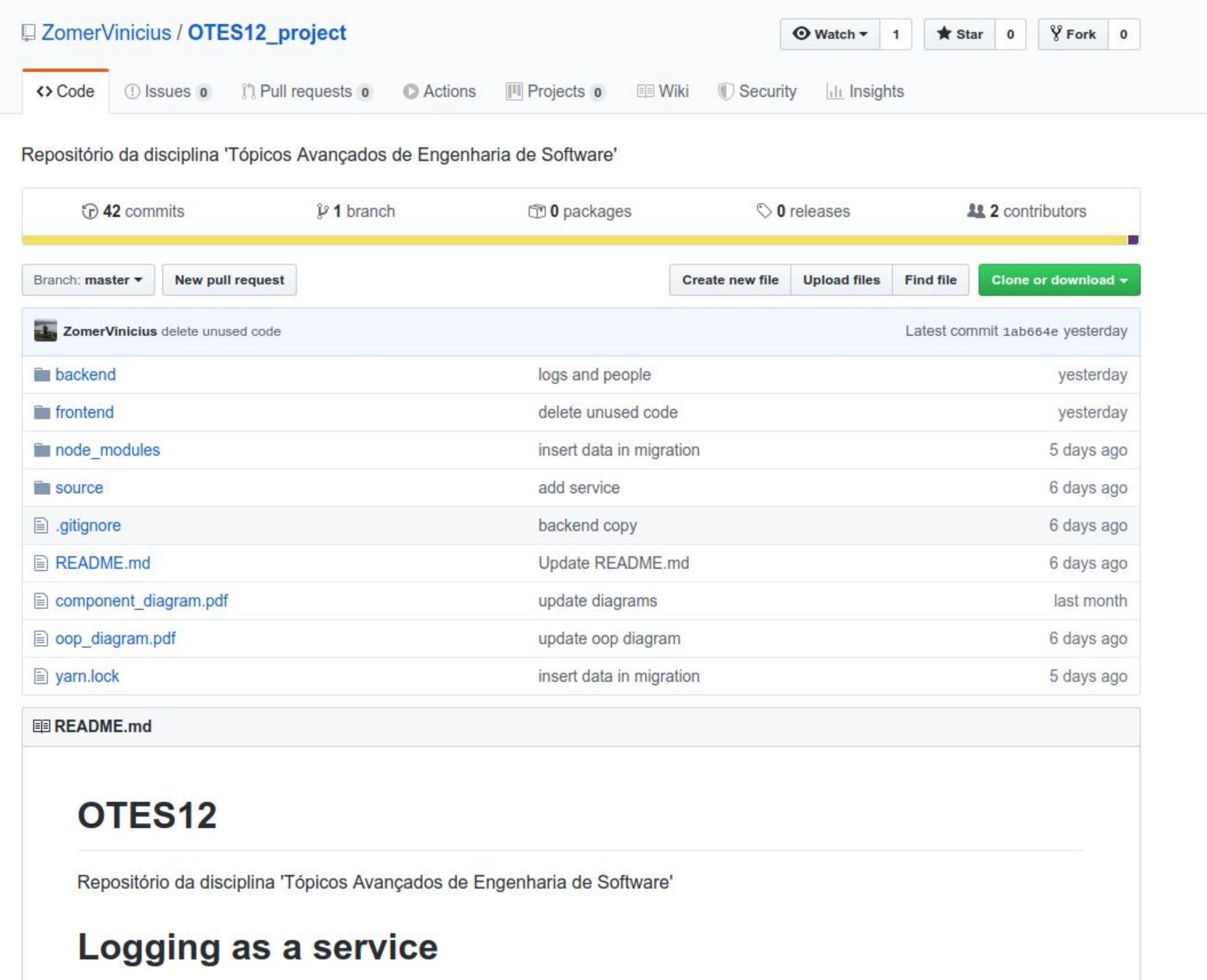
```html
<div>
  <h4>Salvar backup</h4>
  <form action="/create_backup">
    <input type="submit" value="Salvar">
  </form>
</div>
```



```ruby
# BackupsController
def create
  result = backup_service_client.create_backup

  render json: result.to_json
end
```

Backups - Demonstração

https://salty-cove-05656.herokuapp.com/backups



Repositório da disciplina 'Tópicos Avançados de Engenharia de Software'

42 commits | 1 branch | 0 packages | 0 releases | 2 contributors

ZomerVinicius delete unused code — Latest commit 1ab664e yesterday

backend	logs and people	yesterday
frontend	delete unused code	yesterday
node_modules	insert data in migration	5 days ago
source	add service	6 days ago
.gitignore	backend copy	6 days ago
README.md	Update README.md	6 days ago
component_diagram.pdf	update diagrams	last month
oop_diagram.pdf	update oop diagram	6 days ago
yarn.lock	insert data in migration	5 days ago

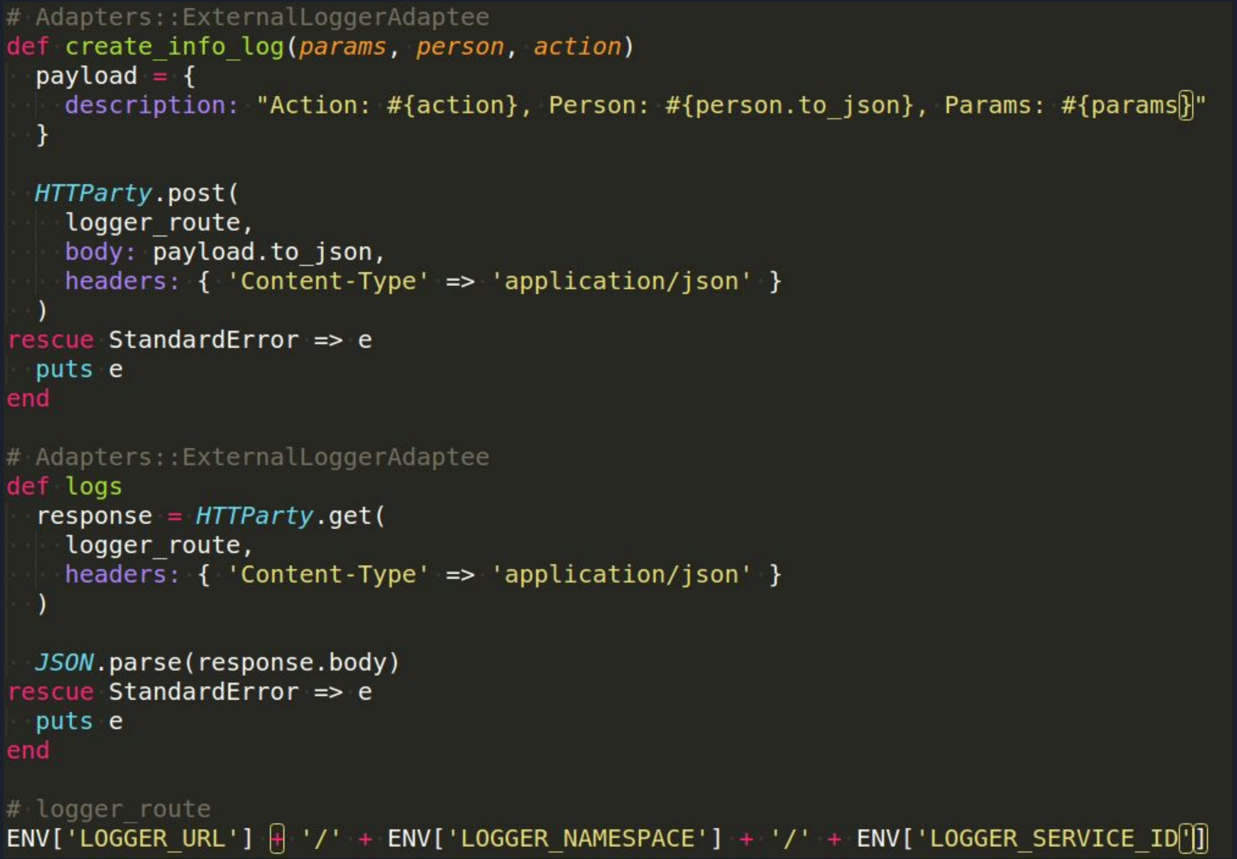
README.md

OTES12

Repositório da disciplina 'Tópicos Avançados de Engenharia de Software'

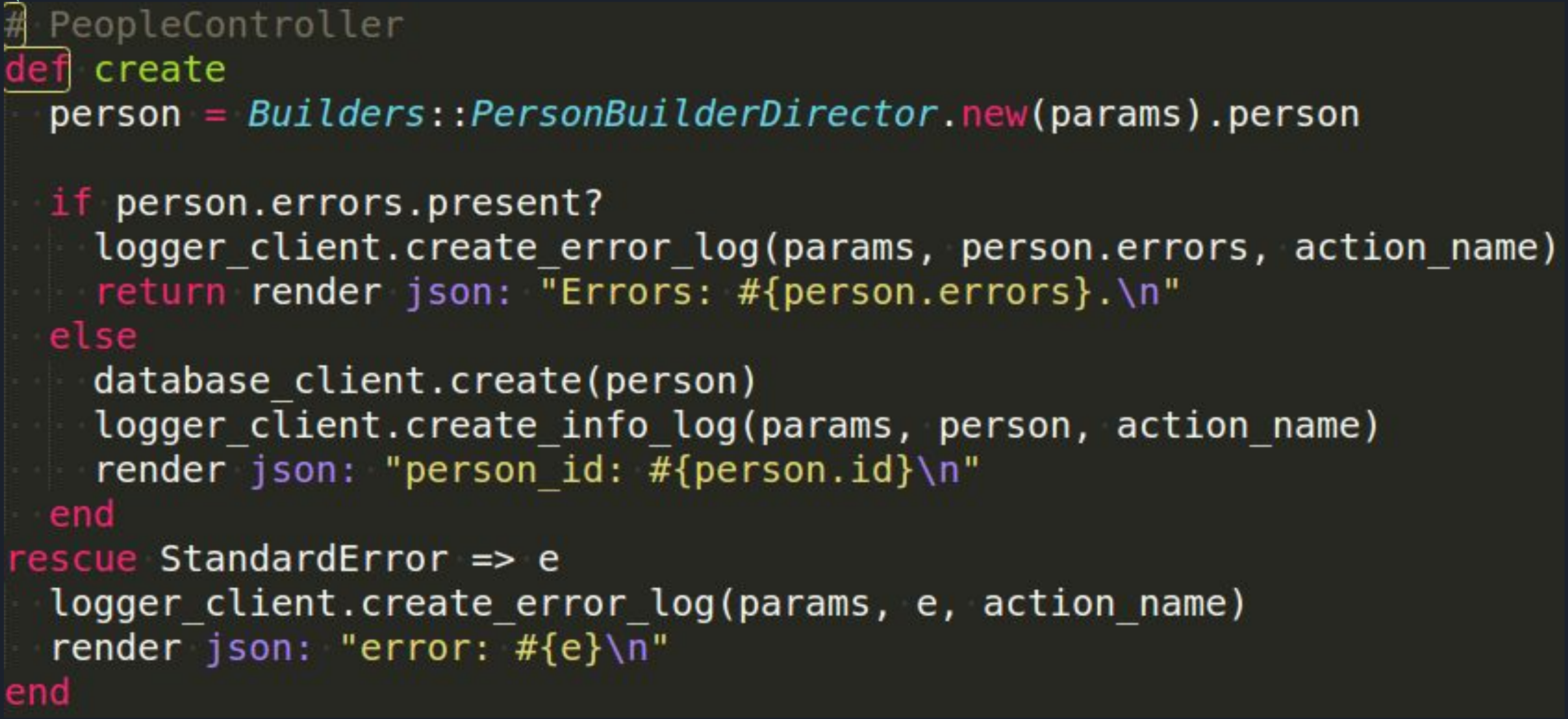
Logging as a service

Logs - Código (chamada API)



```ruby
# Adapters::ExternalLoggerAdaptee
def create_info_log(params, person, action)
  payload = {
    description: "Action: #{action}, Person: #{person.to_json}, Params: #{params}"
  }

  HTTParty.post(
    logger_route,
    body: payload.to_json,
    headers: { 'Content-Type' => 'application/json' }
  )
rescue StandardError => e
  puts e
end

# Adapters::ExternalLoggerAdaptee
def logs
  response = HTTParty.get(
    logger_route,
    headers: { 'Content-Type' => 'application/json' }
  )

  JSON.parse(response.body)
rescue StandardError => e
  puts e
end

# logger_route
ENV['LOGGER_URL'] + '/' + ENV['LOGGER_NAMESPACE'] + '/' + ENV['LOGGER_SERVICE_ID']
```

Logs - Código (uso)

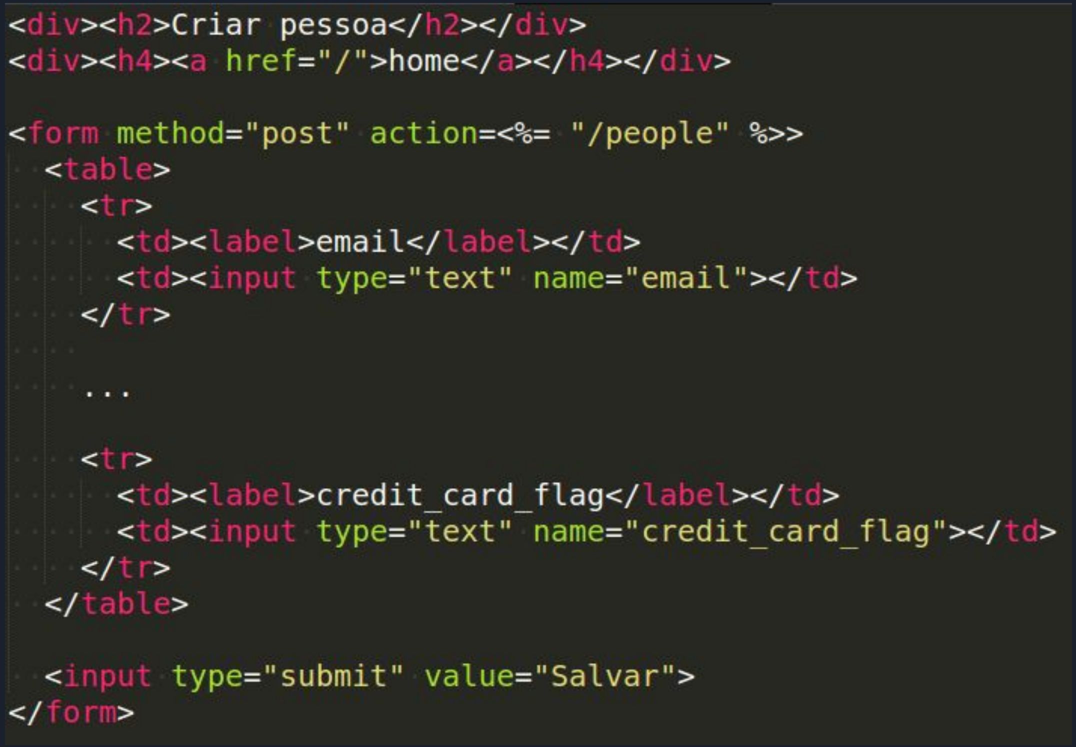


```ruby
# PeopleController
def create
  person = Builders::PersonBuilderDirector.new(params).person

  if person.errors.present?
    logger_client.create_error_log(params, person.errors, action_name)
    return render json: "Errors: #{person.errors}.\n"
  else
    database_client.create(person)
    logger_client.create_info_log(params, person, action_name)
    render json: "person_id: #{person.id}\n"
  end
rescue StandardError => e
  logger_client.create_error_log(params, e, action_name)
  render json: "error: #{e}\n"
end
```

Logs - Demonstração

https://salty-cove-05656.herokuapp.com/logs

Interface de Pessoas - Código

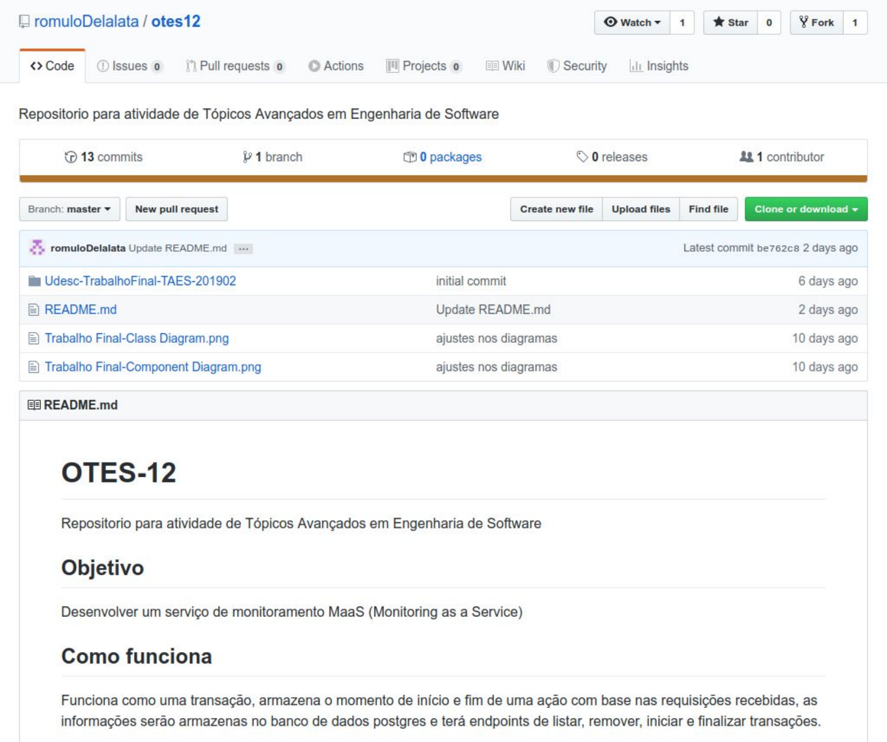


```
<div><h2>Criar pessoa</h2></div>
<div><h4><a href="/">home</a></h4></div>

<form method="post" action=<%= "/people" %>>
  <table>
    <tr>
      <td><label>email</label></td>
      <td><input type="text" name="email"></td>
    </tr>

    ...

    <tr>
      <td><label>credit_card_flag</label></td>
      <td><input type="text" name="credit_card_flag"></td>
    </tr>
  </table>

  <input type="submit" value="Salvar">
</form>
```

Interface de Pessoas - Demonstração

https://salty-cove-05656.herokuapp.com/person_interface

Estatísticas



Repositorio para atividade de Tópicos Avançados em Engenharia de Software

13 commits · 1 branch · 0 packages · 0 releases · 1 contributor

romuloDelalata Update README.md … Latest commit be762c8 2 days ago

Udesc-TrabalhoFinal-TAES-201902	initial commit	6 days ago
README.md	Update README.md	2 days ago
Trabalho Final-Class Diagram.png	ajustes nos diagramas	10 days ago
Trabalho Final-Component Diagram.png	ajustes nos diagramas	10 days ago

README.md

OTES-12

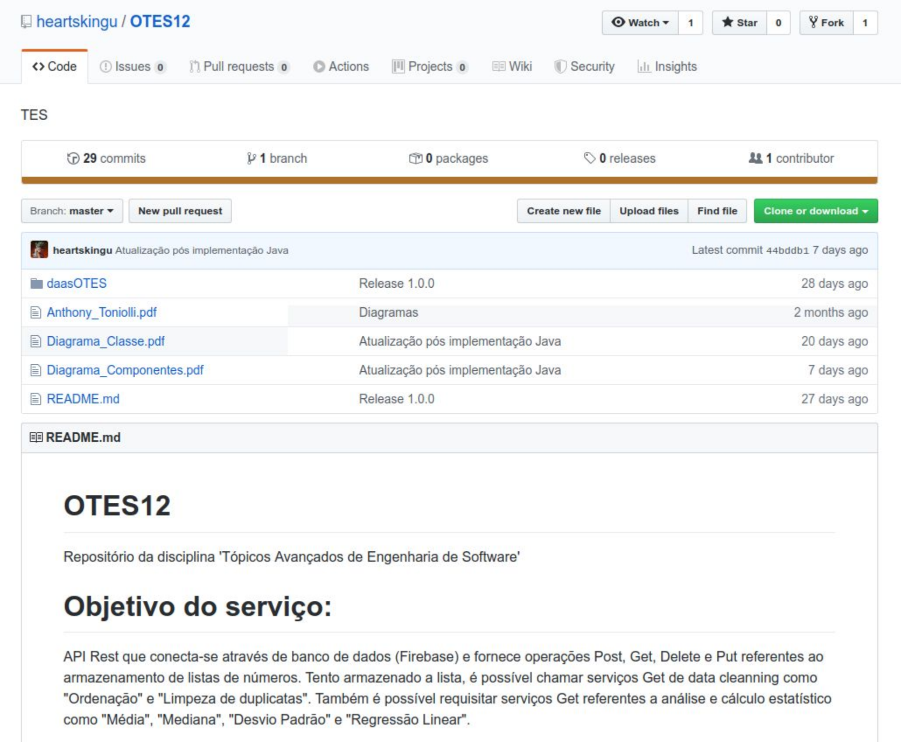
Repositorio para atividade de Tópicos Avançados em Engenharia de Software

Objetivo

Desenvolver um serviço de monitoramento MaaS (Monitoring as a Service)

Como funciona

Funciona como uma transação, armazena o momento de início e fim de uma ação com base nas requisições recebidas, as informações serão armazenas no banco de dados postgres e terá endpoints de listar, remover, iniciar e finalizar transações.



TES

29 commits · 1 branch · 0 packages · 0 releases · 1 contributor

heartskingu Atualização pós implementação Java Latest commit 44bddb1 7 days ago

daasOTES	Release 1.0.0	28 days ago
Anthony_Toniolli.pdf	Diagramas	2 months ago
Diagrama_Classe.pdf	Atualização pós implementação Java	20 days ago
Diagrama_Componentes.pdf	Atualização pós implementação Java	7 days ago
README.md	Release 1.0.0	27 days ago

README.md

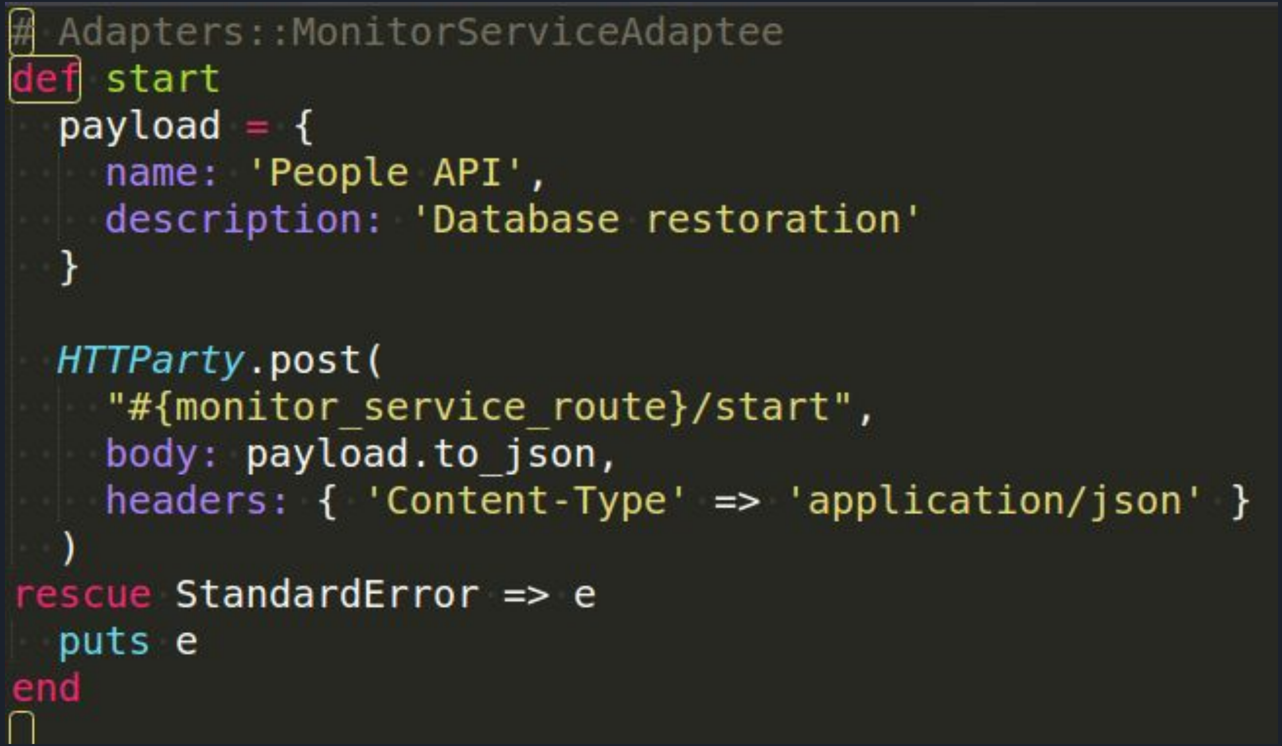
OTES12

Repositório da disciplina 'Tópicos Avançados de Engenharia de Software'

Objetivo do serviço:

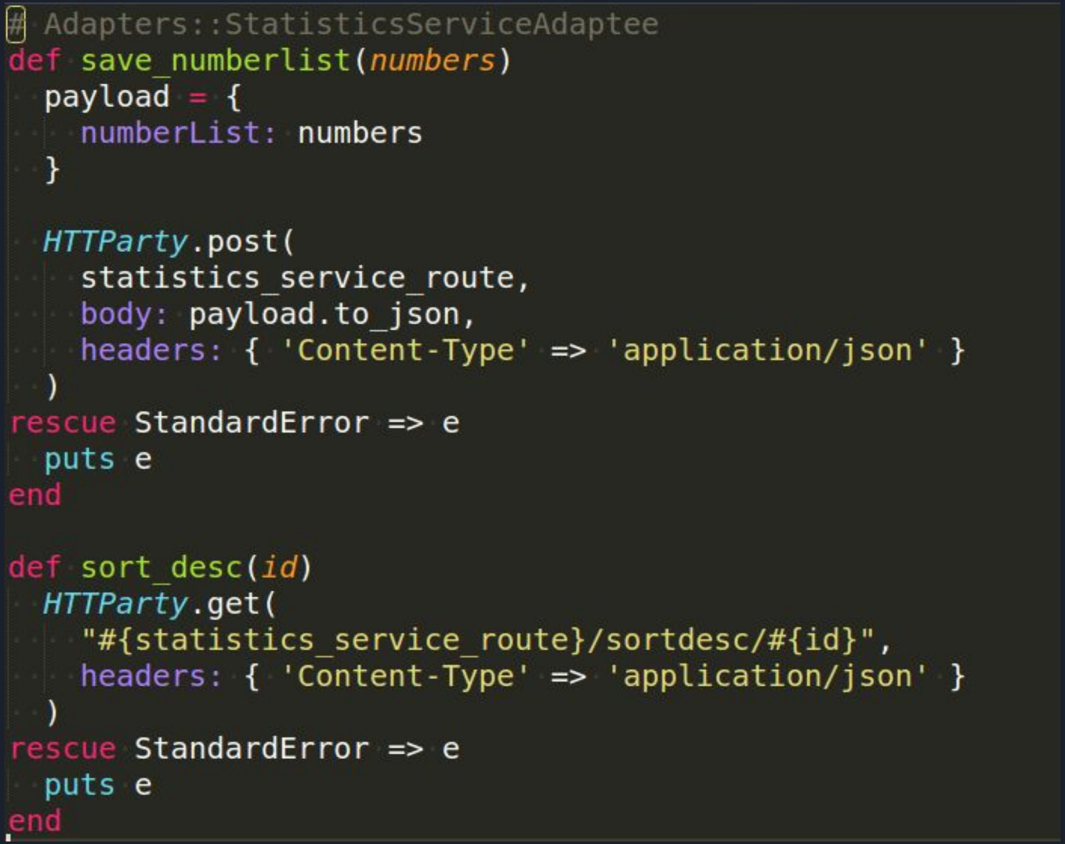
API Rest que conecta-se através de banco de dados (Firebase) e fornece operações Post, Get, Delete e Put referentes ao armazenamento de listas de números. Tento armazenado a lista, é possível chamar serviços Get de data cleanning como "Ordenação" e "Limpeza de duplicatas". Também é possível requisitar serviços Get referentes a análise e cálculo estatístico como "Média", "Mediana", "Desvio Padrão" e "Regressão Linear".

Estatísticas - Código (chamada API maas)



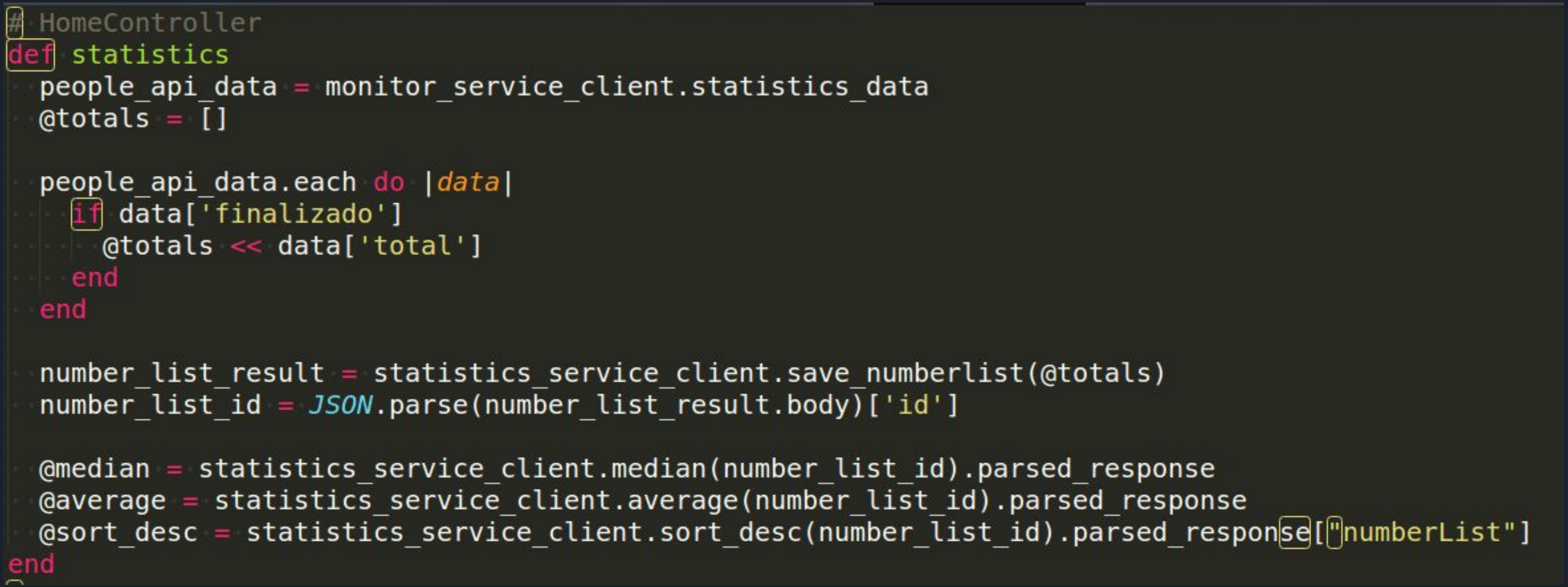
```ruby
# Adapters::MonitorServiceAdaptee
def start
  payload = {
    name: 'People API',
    description: 'Database restoration'
  }

  HTTParty.post(
    "#{monitor_service_route}/start",
    body: payload.to_json,
    headers: { 'Content-Type' => 'application/json' }
  )
rescue StandardError => e
  puts e
end
```

Estatísticas - Código (chamada API)



```ruby
# Adapters::StatisticsServiceAdaptee
def save_numberlist(numbers)
  payload = {
    numberList: numbers
  }

  HTTParty.post(
    statistics_service_route,
    body: payload.to_json,
    headers: { 'Content-Type' => 'application/json' }
  )
rescue StandardError => e
  puts e
end

def sort_desc(id)
  HTTParty.get(
    "#{statistics_service_route}/sortdesc/#{id}",
    headers: { 'Content-Type' => 'application/json' }
  )
rescue StandardError => e
  puts e
end
```

Estatísticas - Uso

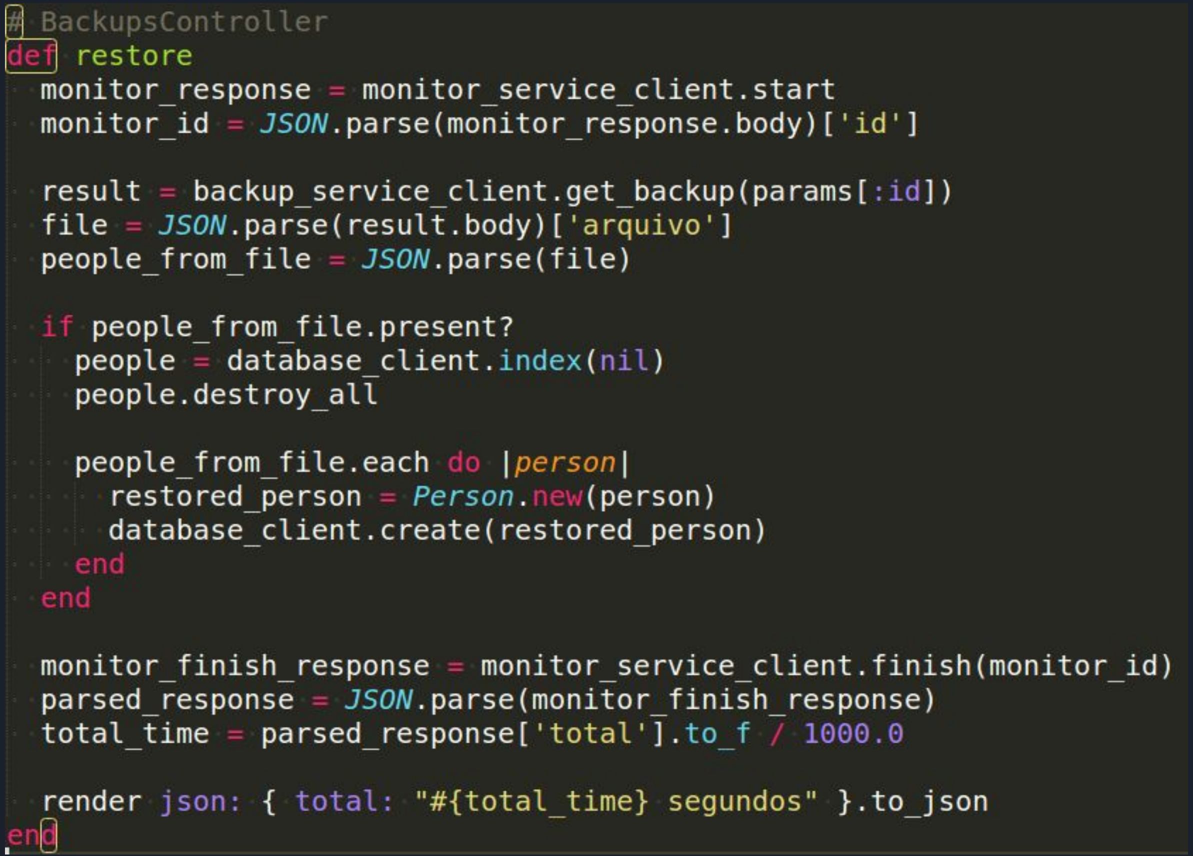


```
# HomeController
def statistics
  people_api_data = monitor_service_client.statistics_data
  @totals = []

  people_api_data.each do |data|
    if data['finalizado']
      @totals << data['total']
    end
  end

  number_list_result = statistics_service_client.save_numberlist(@totals)
  number_list_id = JSON.parse(number_list_result.body)['id']

  @median = statistics_service_client.median(number_list_id).parsed_response
  @average = statistics_service_client.average(number_list_id).parsed_response
  @sort_desc = statistics_service_client.sort_desc(number_list_id).parsed_response["numberList"]
end
```

Estatísticas - Demonstração

https://salty-cove-05656.herokuapp.com/statistics

Backups - Código (uso)



```ruby
# BackupsController
def restore
  monitor_response = monitor_service_client.start
  monitor_id = JSON.parse(monitor_response.body)['id']

  result = backup_service_client.get_backup(params[:id])
  file = JSON.parse(result.body)['arquivo']
  people_from_file = JSON.parse(file)

  if people_from_file.present?
    people = database_client.index(nil)
    people.destroy_all

    people_from_file.each do |person|
      restored_person = Person.new(person)
      database_client.create(restored_person)
    end
  end

  monitor_finish_response = monitor_service_client.finish(monitor_id)
  parsed_response = JSON.parse(monitor_finish_response)
  total_time = parsed_response['total'].to_f / 1000.0

  render json: { total: "#{total_time} segundos" }.to_json
end
```

Docs

https://github.com/pedroCervi/OTES12/tree/master/docs