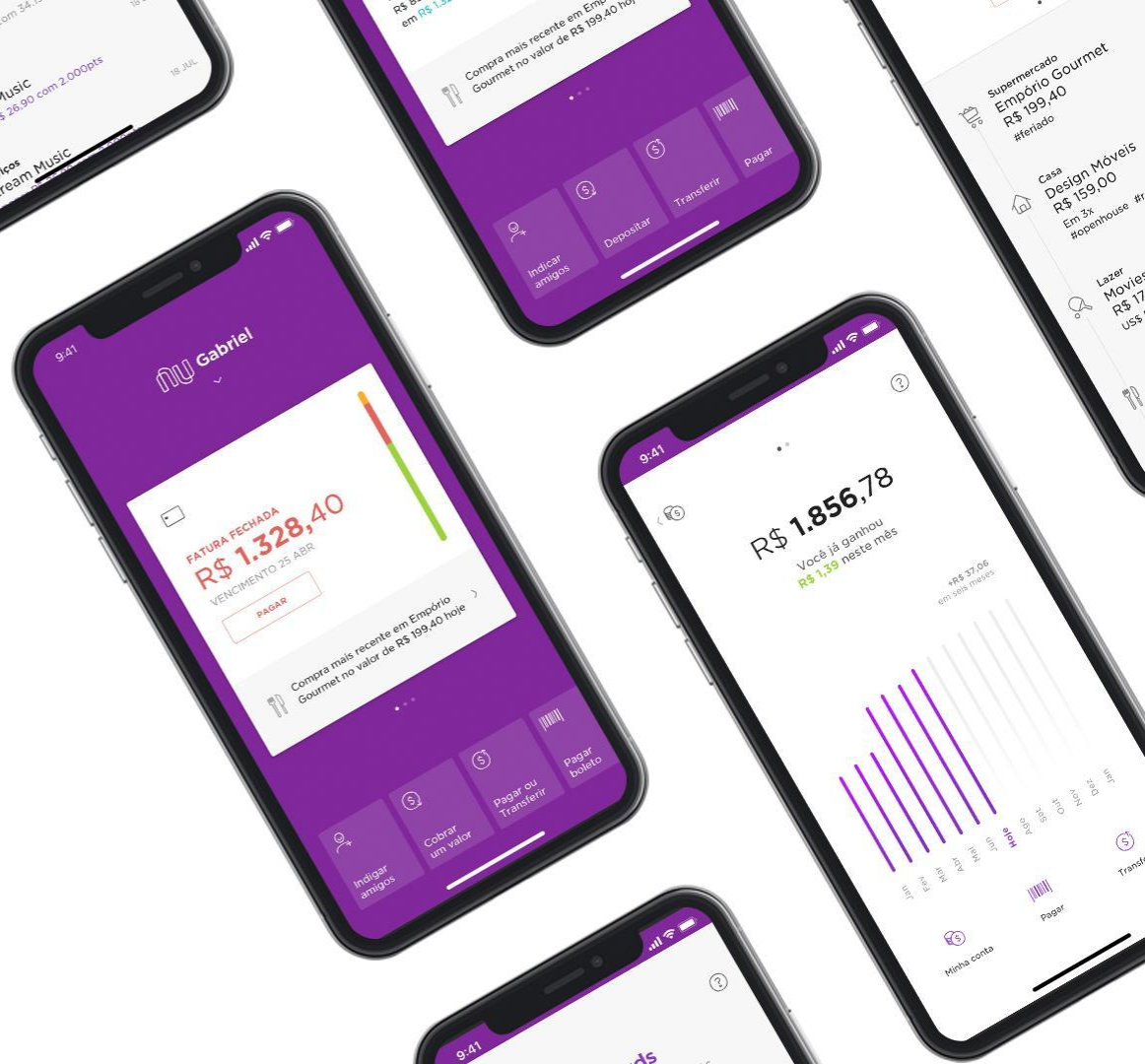
nubank

Clojure: do zero ao deploy de uma API REST



Quem somos nós

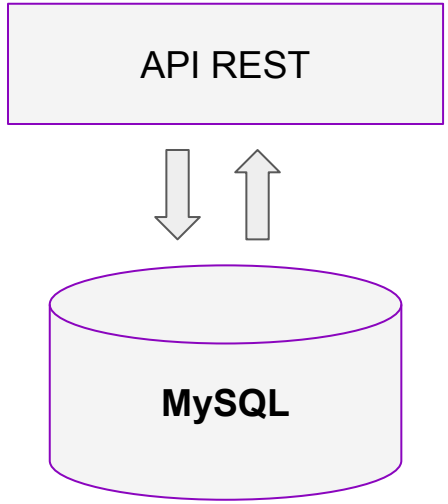
Ana Luisa Bavati
Software Engineer

Newton Beck
Software Engineer

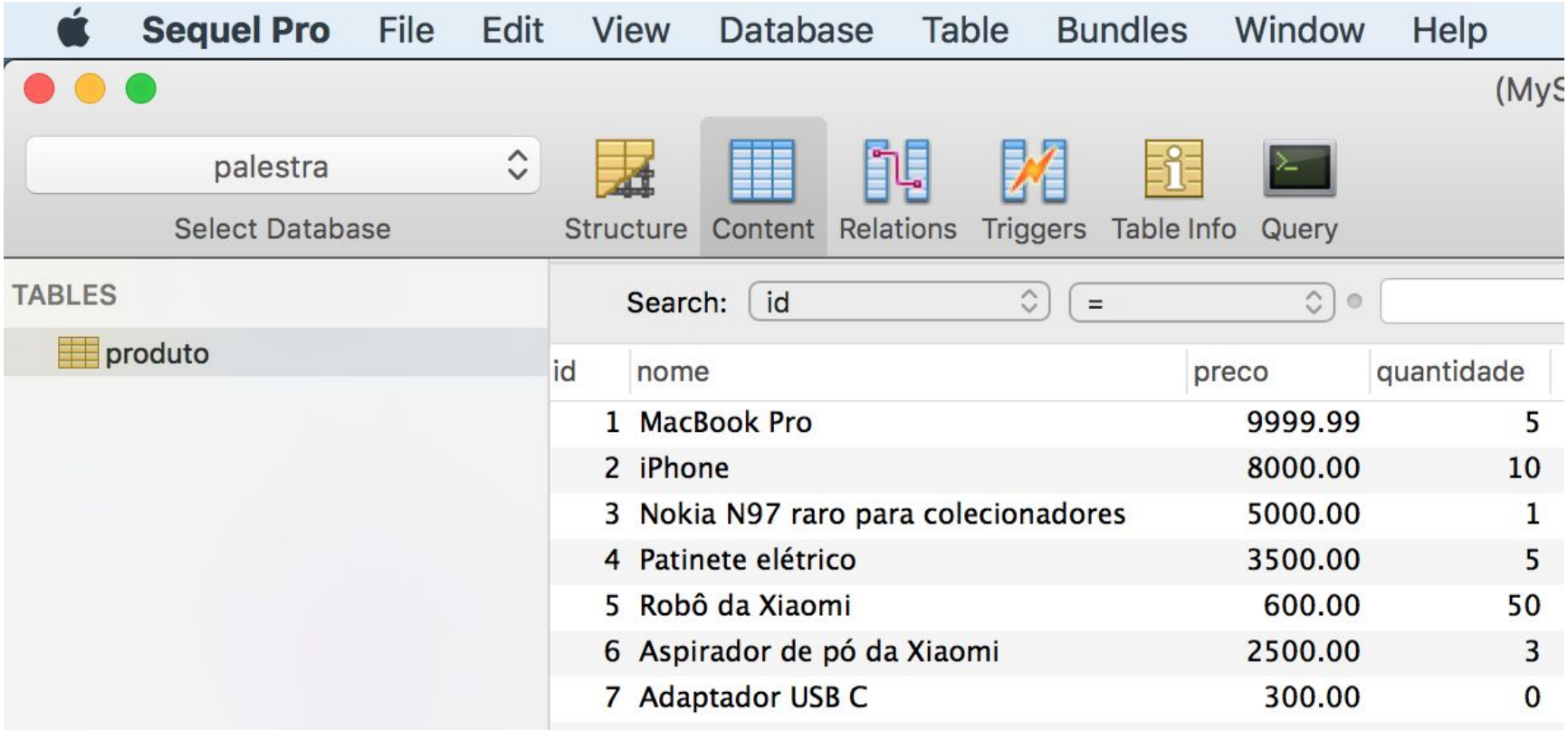
O que vamos fazer

Vamos criar uma API REST com Clojure

- Criaremos um endpoint GET para listar os produtos disponíveis do nosso banco de dados MySQL



O que temos no nosso banco de dados?



O que vamos retornar na API?

```
curl http://localhost:3000/produtos-disponiveis
[{"id":1,"nome":"MacBook Pro","preco":9999.99,"quantidade":5},
{"id":2,"nome":"iPhone","preco":8000.00,"quantidade":10},
{"id":3,"nome":"Nokia N97 raro para colecionadores","preco":5000.00,"quantidade":1},
{"id":4,"nome":"Patinete elétrico","preco":3500.00,"quantidade":5},
{"id":5,"nome":"Robô da Xiaomi","preco":600.00,"quantidade":50},
{"id":6,"nome":"Aspirador de pó da Xiaomi","preco":2500.00,"quantidade":3}]
```

Clojure

O que é Clojure?

Clojure é um dialeto de LISP que roda em cima da **JVM**.

Como funciona?

```
(operador operando-1 operando-2 operando-3 ...)
```

Como funciona uma declaração de variável?

```
;(def nome-da-variavel valor-da-variavel)

(def nome "Ana")
(def idade 24)
(def ja-palestrou? true)
```

Como declaramos um mapa em uma variável?

```
;(def nome-da-variavel
;  {:chave  valor
;   :chave2 valor2})

(def pessoa
  {:nome          "Ana"
   :idade          24
   :ja-palestrou? true})
```

Como funciona uma chamada de função?

```clojure
(def nome "Ana")

;(nome-da-funcao parametro-1 parametro-2)

(println "Ola" nome)
(+ 1 2)
```

Como funciona uma comparação?

```
(def idade 24)

;(> valor-1 valor-2)

(> idade 18)
(= idade 18)
(< idade 18)
```

Como funciona um if/else?

```
;(if (condicao)
;    (corpo-se-for-true)
;    (corpo-se-for-false))

(if (< idade 65)
  (println "Não é idoso")
  (println "É idoso"))
```

Como funciona uma declaração de função?

```
;(defn nome-da-funcao
;      [parametro1 parametro2]
;      (corpo-da-funcao))

(defn diz-se-eh-idoso [idade]
  (if (< idade 65)
    (println "Não é idoso")
    (println "É idoso")))
```

Como declaramos uma variável local?

```
(defn diz-nome-completo [nome sobrenome]
  (let [nome-completo (str nome sobrenome)]
    (println nome-completo)))
```

Como agrupamos as funções do nosso código?

```
; (ns projeto.pasta.arquivo)

(ns palestra-clojure.logic.produto)
; várias funções

(ns palestra-clojure.logic.controller)
; várias funções
```

Como importamos um namespace?

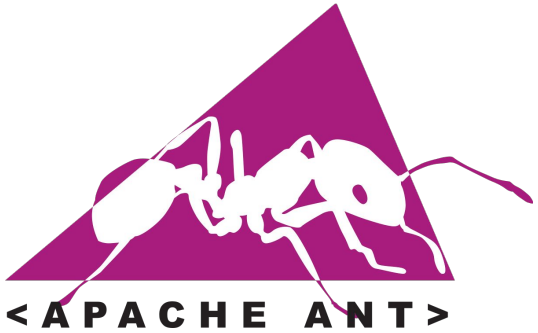
```clojure
; (ns projeto.pasta.arquivo)

(ns palestra-clojure.logic.produto)
; várias funções

(ns palestra-clojure.logic.controller
  (:require [palestra-clojure.logic.produto :as logic]))
; várias funções
```

Criando um projeto

Antes, uma pergunta

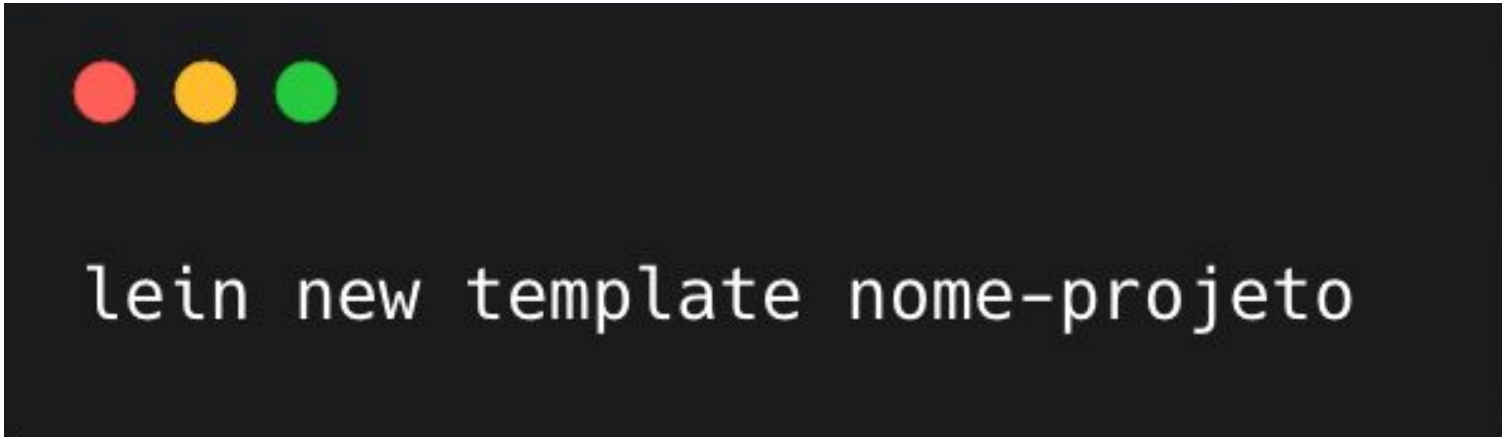
- Qual ferramenta de **construção** e **gerenciamento de dependências** vocês usam no projeto Java de vocês?

Algumas ferramentas de construção em Java



Vamos usar o Leiningen no nosso projeto

Como criamos nosso projeto com Leiningen?



Estrutura de diretório

```
| .gitignore
| doc
| | intro.md
❶ | project.clj
| README.md
❷ | resources
| src
| | nome_do_projeto
❸ | | | core.clj
❹ | test
| | nome_do_projeto
| | | core_test.clj
```

Composição inicial do projeto:

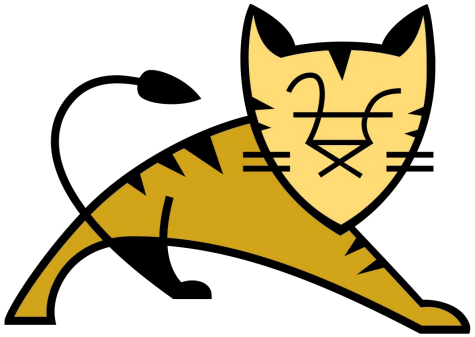
- **project.clj:** definição de build
- **/src:** diretório de source, com um arquivo inicial core.clj
- **/test:** diretório de teste, com um arquivo inicial core_test.clj

Servidor HTTP

Antes, algumas perguntas

- Qual o **servidor HTTP** que vocês usam pra rodar o projeto Java de vocês?
- Qual o **framework** vocês usam para desenvolver Java para a web?

Alguns servidores Java



Alguns frameworks web em Java

Vamos usar pedestal no nosso projeto

http://pedestal.io/

Como adicionamos pedestal no nosso projeto?

```
(defproject palestra-clojure "0.1.0-SNAPSHOT"
  :description "FIXME: write description"
  :url "http://example.com/FIXME"
  :license {:name "Eclipse Public License"
            :url  "http://www.eclipse.org/legal/epl-v10.html"}
  :dependencies [[org.clojure/clojure "1.8.0"]
                 [io.pedestal/pedestal.service "0.5.1"]
                 [io.pedestal/pedestal.route   "0.5.1"]
                 [io.pedestal/pedestal.jetty   "0.5.1"]
                 [org.clojure/data.json        "0.2.6"]
                 [org.slf4j/slf4j-simple       "1.7.21"]]
  :main palestra-clojure.main)
```

Como rodamos nosso servidor HTTP?

```clojure
(ns palestra-clojure.main
  (:gen-class)
  (:require [io.pedestal.http :as http]
            [palestra-clojure.service :as service]))

(defn cria-o-servidor []
  (http/create-server
    {::http/routes  service/rotas
     ::http/type    :jetty
     ::http/port    3000}))

(defn -main []
  (http/start (cria-o-servidor)))
```

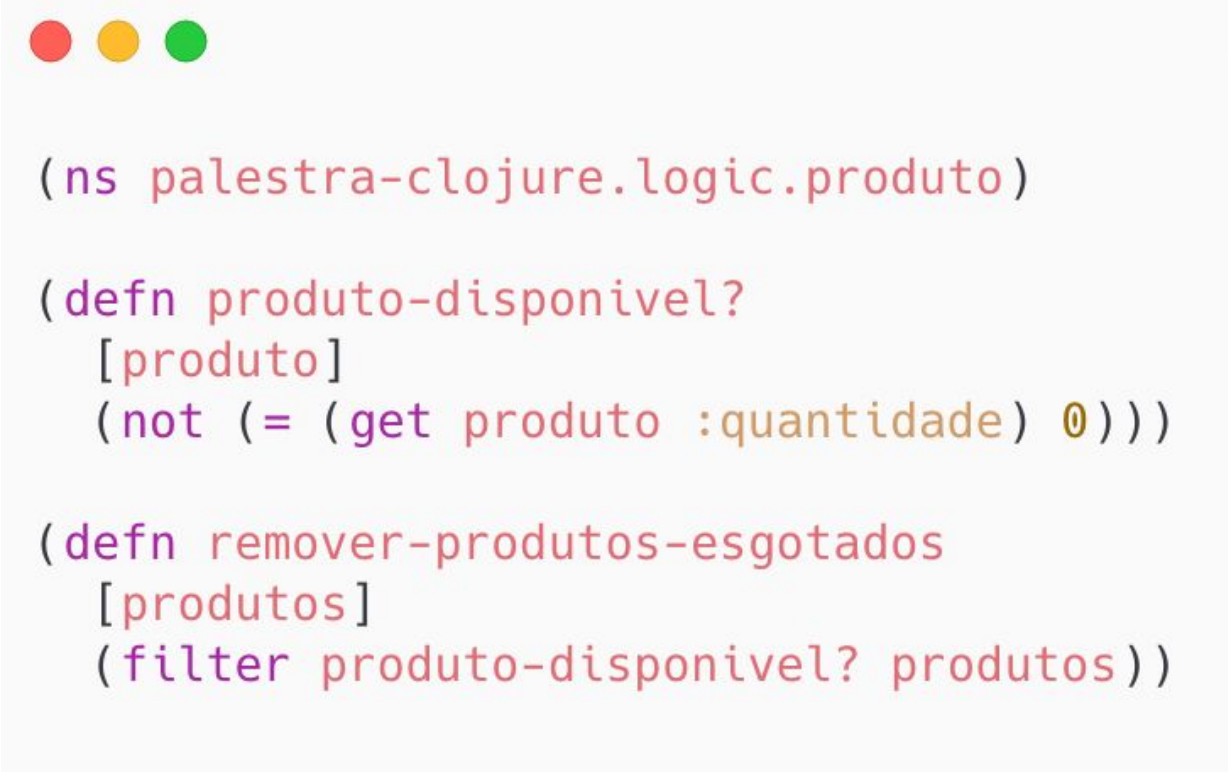
Como registramos nossas rotas?

```clojure
(ns palestra-clojure.service
  (:gen-class)
  (:require [io.pedestal.http.route.definition.table :as table]
            [palestra-clojure.controllers.produto :as controller]))

(defn listar-produtos-disponiveis [request]
  {:status  200
   :body    (controller/listar-produtos-disponiveis)
   :headers {"Content-Type" "application/json"}})

(def rotas
  (table/table-routes
   [["/produtos-disponiveis" :get listar-produtos-disponiveis :route-name :listar-produtos-
disponiveis]]))
```

Como implementamos nosso controller?

```clojure
(ns palestra-clojure.controllers.produto
  (:require [palestra-clojure.logic.produto :as logic]
            [palestra-clojure.db.produto :as db]))

(defn listar-produtos-disponiveis []
  (let [produtos (db/buscar)]
    (logic/remover-produtos-esgotados produtos)))
```

Como implementamos nossa lógica?



```clojure
(ns palestra-clojure.logic.produto)

(defn produto-disponivel?
  [produto]
  (not (= (get produto :quantidade) 0)))

(defn remover-produtos-esgotados
  [produtos]
  (filter produto-disponivel? produtos))
```

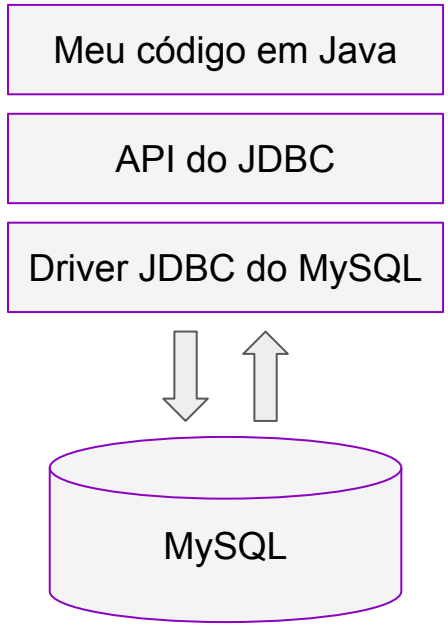
Acessando o banco

Antes, uma pergunta

Como vocês acessam banco de dados em Java?

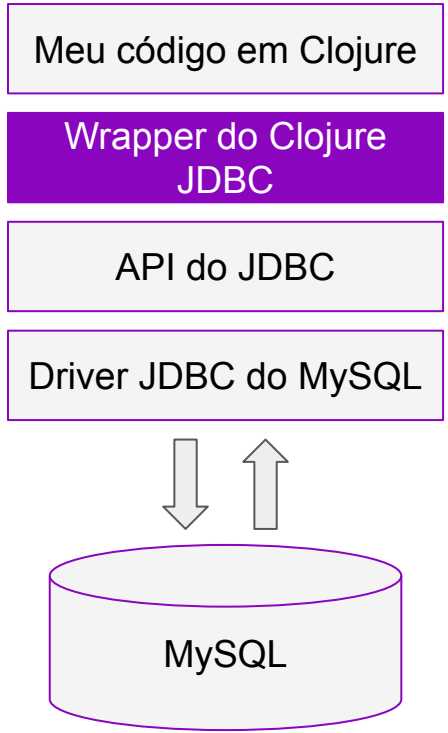
Como acessamos um banco de dados em Java

1. Adicionamos a biblioteca do driver do banco de dados no nosso projeto
2. Usamos a API do JDBC para acessar nosso banco de dados



Como acessamos um banco de dados em Clojure

1. Adicionamos a biblioteca do driver do banco de dados no nosso projeto
2. Adicionamos a API Clojure do JDBC para acessar nosso banco de dados
3. Usamos a API Clojure do JDBC para acessar o banco de dados
   https://github.com/clojure/java.jdbc



Adicionando as bibliotecas no nosso projeto

```
(defproject palestra-clojure "0.1.0-SNAPSHOT"
  :description "FIXME: write description"
  :url "http://example.com/FIXME"
  :license {:name "Eclipse Public License"
            :url  "http://www.eclipse.org/legal/epl-v10.html"}
  :dependencies [[org.clojure/clojure "1.8.0"]
                 [io.pedestal/pedestal.service "0.5.1"]
                 [io.pedestal/pedestal.route   "0.5.1"]
                 [io.pedestal/pedestal.jetty   "0.5.1"]
                 [org.clojure/data.json        "0.2.6"]
                 [org.slf4j/slf4j-simple       "1.7.21"]
                 [mysql/mysql-connector-java "5.1.47"]
                 [org.clojure/java.jdbc "0.7.9"]]
  :main palestra-clojure.main)
```

Usando o wrapper Clojure de JDBC

```
;(jdbc/query dados-de-conexão-com-o-banco [consulta-sql parâmetro-1 parâmetro-2])

(def dados-de-conexao {
  :dbtype "mysql"
  :dbname "palestra"
  :user "root"
  :password "123"
  :host "localhost"
  :port 3306
})

(jdbc/query dados-de-conexao ["select * from produto"])
```

O que a query retorna para o nosso código?

```
({:id 1, :nome "MacBook Pro", :preco 9999.99M, :quantidade 5}
 {:id 2, :nome "iPhone", :preco 8000.00M, :quantidade 10}
 {:id 3, :nome "Nokia N97 raro para colecionadores", :preco 5000.00M, :quantidade 1}
 {:id 4, :nome "Patinete elétrico", :preco 3500.00M, :quantidade 5}
 {:id 5, :nome "Robô da Xiaomi", :preco 600.00M, :quantidade 50}
 {:id 6, :nome "Aspirador de pó da Xiaomi", :preco 2500.00M, :quantidade 3})
```

Voltando ao nosso controller

```clojure
(ns palestra-clojure.controllers.produto
  (:require [palestra-clojure.logic.produto :as logic-produto]
            [palestra-clojure.db.produto :as db-produto]))

(defn listar-produtos-disponiveis []
  (let [produtos (db-produto/buscar)]
    (logic-produto/remover-produtos-esgotados produtos)))
```

Deploy do projeto

Antes, algumas perguntas

- Qual é o formato de arquivo que vocês usam para fazer o deploy de um projeto Java?

Como distribuímos código Java?

Podemos empacotar nosso código num arquivo **.jar** ou **.war** para fazer seu deploy

Como distribuímos código Clojure?

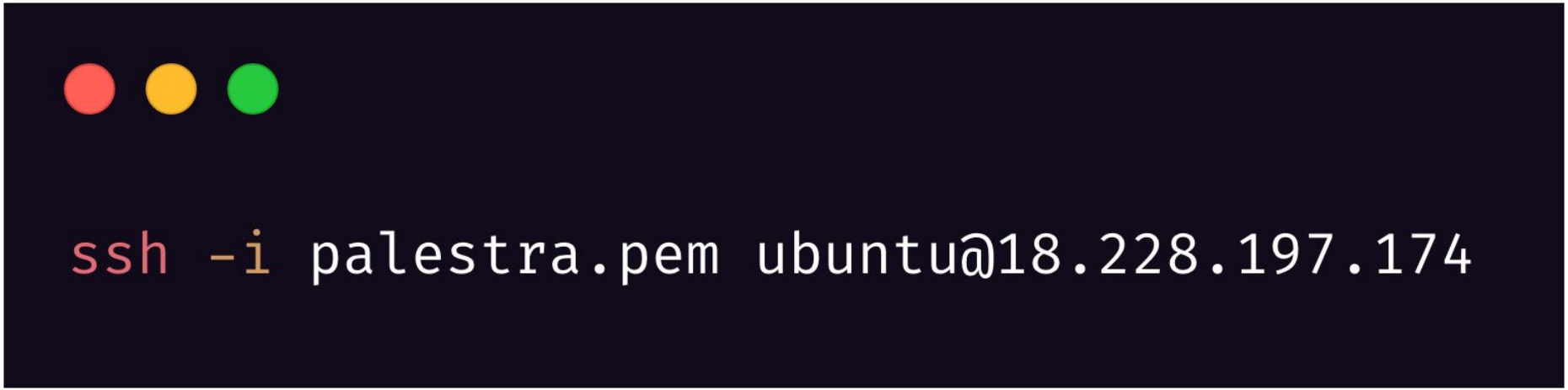
Podemos empacotar nosso código num arquivo **.jar** ou **.war** para fazer seu deploy, assim como fazemos com Java

Gerando um arquivo .jar para a distribuição?

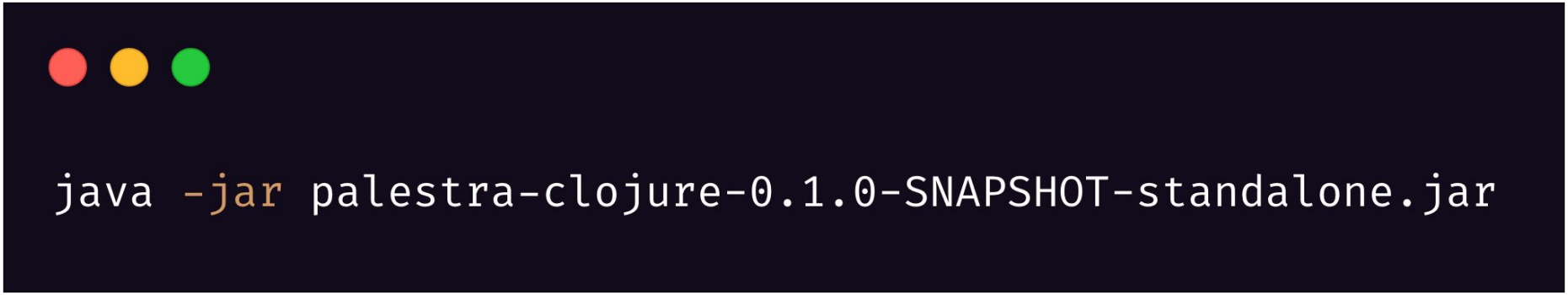
Copiando o arquivo .jar para o servidor

```
scp \
  -i palestra.pem \
  target/palestra-clojure-0.1.0-SNAPSHOT.jar \
  ubuntu@18.228.197.174:~/
```

Acessando no servidor



Executando o arquivo .jar



Acessando a API

https://bit.ly/2xPJpXb

Para saber mais

Repositório do projeto e próximos passo

https://bit.ly/2XWiDqo

Clojure South

- **Data**: 31 de agosto e 1 setembro
- **Local**: Nubank
- **Informações**: https://clojure-south.com/

Estamos contratando!

sou.nu/jobs-at-nubank


bank