Introdução ao NodeJS

express.js + SQL com knex.js

Agenda

- npm
- express.js
- rotas
- verbos HTTP
- req.query versus req.params
- body-parser
- revisão SQL
- knex.js
- knexfile.js
- migrations

NPM

- Gerenciador de pacotes do Node
- Muito além do node *(mas não falaremos disso aqui)*
- Cria projetos/módulos (ex: npm init)
- Instala módulos (ex: npm instal linebyline; sudo npm -g install nodemon)
- https://www.npmjs.com/

Express.js

- Framework de roteamento
- Mantido pela StrongLoop (comprada recentemente pela IBM)
- Padrão *de facto* de mercado
- **var express = require(“express”); var app = express();**
- Conteúdo estático (ex: **app.use(express.static("www"));**)
- http://expressjs.com/en/starter/hello-world.html

Rotas

- Endereços atendidos pelo app
- Componente Router do express para separar logicamente os módulos do app
- Suporte a “expresões regulares” nas rotas
- http://expressjs.com/en/guide/routing.html
- **app.get(“/hellojs”,function(req,res){/* ... */});**

Verbos HTTP

- Do protocolo HTTP
- Foco em GET/POST/PUT/DELETE
- GET/DELETE não tem corpo
- http://expressjs.com/en/4x/api.html#app.METHOD

req.query

- Recebe os parâmetros e os transforma em um objeto javascript
- Disponível em qualquer verbo
- (ex: http://localhost :3000/teste?a=1&b=2
- http://expressjs.com/en/4x/api.html#req.query

req.params

- Uma forma de recuperar valores do caminho
- Preferir parâmetros nomeados
- **(ex: app.get(“/atores/:idator”,function(req,res){console.log(req.params.idator);});)**
- **http://expressjs.com/en/api.html#req.params**

req.body (body-parser)

- Corpo da postagem
- Essencial para aplicativos que trabalharão com json
- Trabalhando com json temos uma aplicação mais simples
- http://expressjs.com/en/api.html#req.body

SQL

- Inventado pela IBM em 1492 por Vasco da Gama *[Citation Needed]*
  - Encomendado por El-Rey de Espanha. Algo assim.
  - https://en.wikipedia.org/wiki/SQL
- Não importa o panorama, só o COBOL e o SQL permanecem
- Evite abstrações demasiadas (Knex em vez de Bookshelf/Sequelize)
- Normalize enquanto é possível
- NUNCA chame a coluna de ID apenas de ID
- Suas Tabelas representam boa parte do negócio a resolver
- Atenção aos *dialetos* do SQL (cada banco tem o seu)
- http://www.w3schools.com/sql/

CREATE TABLE

- create table contato(idcontato integer not null primary key, nomecontato varchar (255) not null)
- create table evento(idevento integer not null primary key,dataevento date not null default now())
- create table contato_evento(idcontato integer not null, idevento integer not null, foreign key (idcontato) references contato(idcontato),foreign key (idevento) references evento(idevento))

INSERT

- insert into contato values(1,’João’);
- insert into contato (nomecontato,idcontato) values (‘Bruna’,2);
- insert into evento (idevento,dataevento) values (1,’2015-12-05’);
- insert into contato_evento (idcontato,idevento) values (1,1);

UPDATE

- update contato set nomecontato = Surfistinha’ where idcontato = 2;

DELETE

- delete from contato where idcontato = 1;

SELECT

- select * from contato;
- select idevento,dataevento from evento;
- select * from evento natural join contato_evento
  - com natural join suas consultas ficam melhores

CREATE VIEW

- create view evtscontatos as select c.nomecontato,e.dataevento from contato c natural join evento e
  - agora podemos faze apenas select * from evtscontatos
  - views são ferramentas fundamentais para uma boa modelagem onde o banco é devidamente respeitado

Knex.js

- Um “query builder”
- Consulta o banco a partir do node
- Ferramentas adicionais (promessas, migrações, perfis, sementes, etc.)
- https://github.com/tgriesser/knex
- Tudo o que há para saber está em http://knexjs.org/
- **npm -g install knex**
- **knex init**

knexfile.js

- Dados de conexão e de perfil
- **var knexfile = require('./knexfile');**
- **var kcfg = typeof process.env.OPENSHIFT_CLOUD_DOMAIN == "undefined" ? knexfile.development : knexfile.production**
- **var knex = require('knex')(kcfg);**
- *A referência ao knex deve ser criada no 'boot' da aplicação (criar um pool é sempre oneroso) e repassado para os módulos que eventualmente precisem de acesso ao banco de dados*
- **knex("contato").select().then(function(ret){console.log(ret);});**
- **knex.raw("select nomecontato from contato").then(function(ret){console.log(ret.rows);});**

migrations

- **knex migrate:make versao_inicial_do_banco**
- Arquivos de migração são módulos que contam com uma função **up** e uma função **down**;
- Teste sempre os seus arquivos de mgiração, todas as versões do banco devem ser reversíveis
- http://pastebin.com/WHaVjpgb
- Tudo o que há para saber está em http://knexjs.org/#Schema-createTable