./SRC/PROJECT

CODERS.BAY

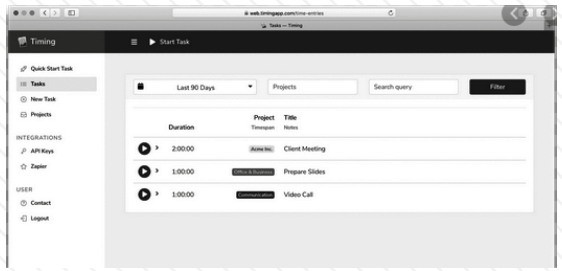
./SRC/PROJECT

./SRC/PROJECT/ROUTE

| PHIL ZIEGELBAUER

CODERS.BAY, 17.06.2021, Linz Austria |

For what the ... ?



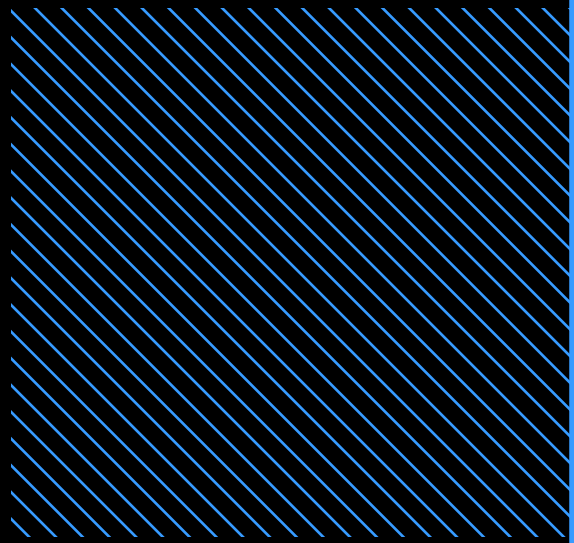
Web Frontend

Server Backend

metachron Database

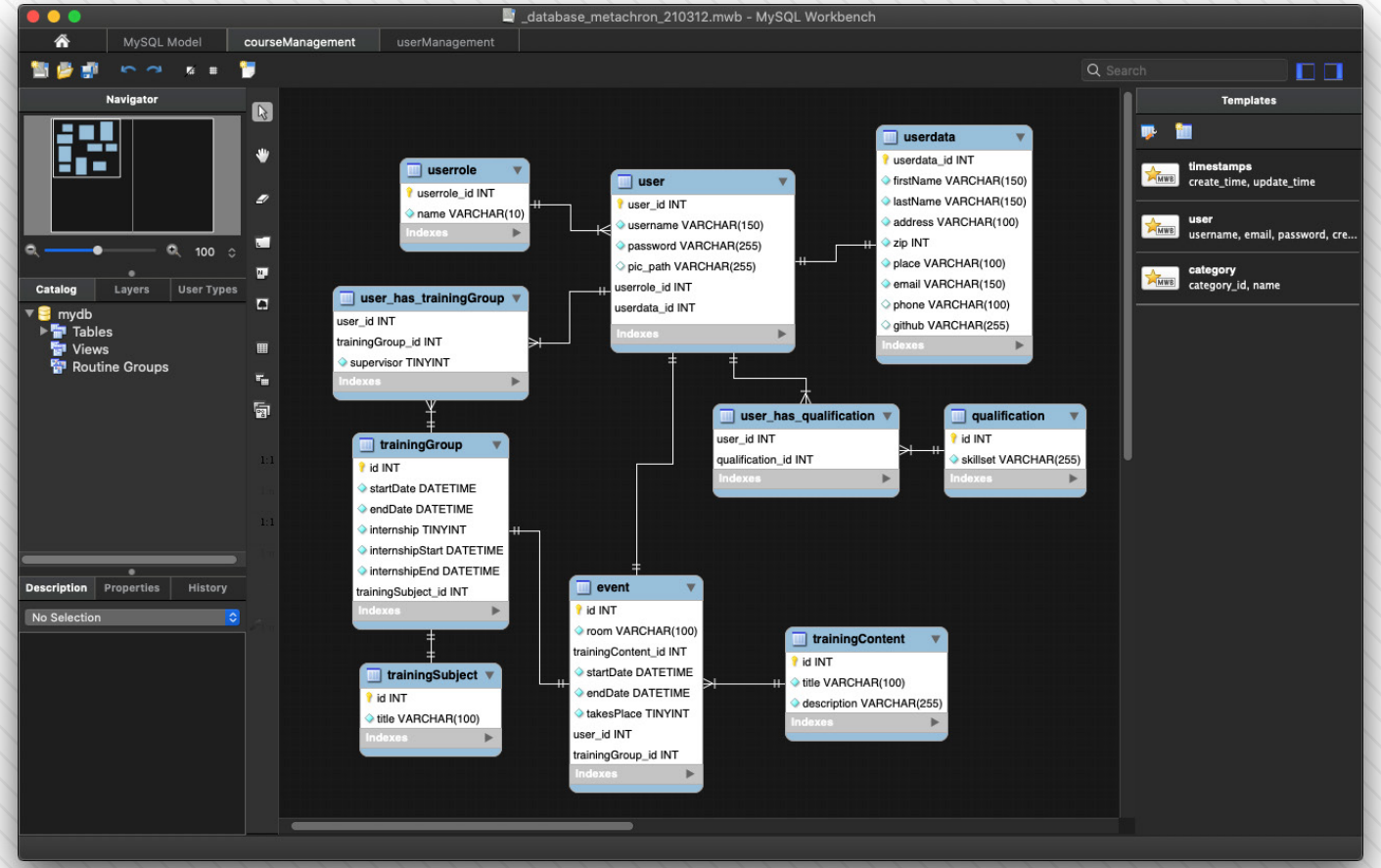
./SRC/PROJECT

./SRC/PROJECT/DATABASE



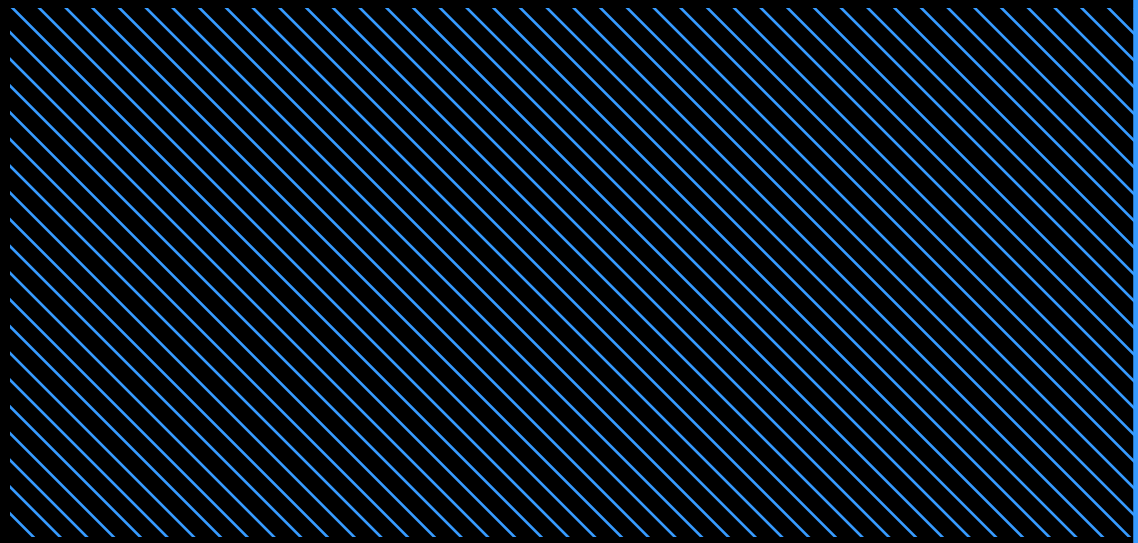
SELECT * FROM?

- MYSQLWORKBENCH
  - database design



./SRC/PROJECT

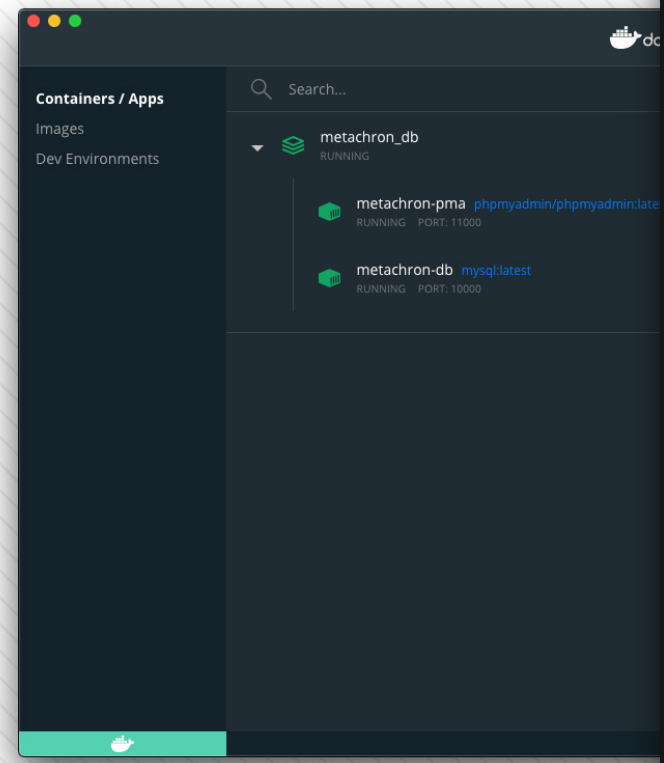
./SRC/PROJECT/DOCKER



CONTAINER != ENOUGH?

- docker desktop
  - setup docker containers
    - pma (port: 11000)
    - db (port: 10000)

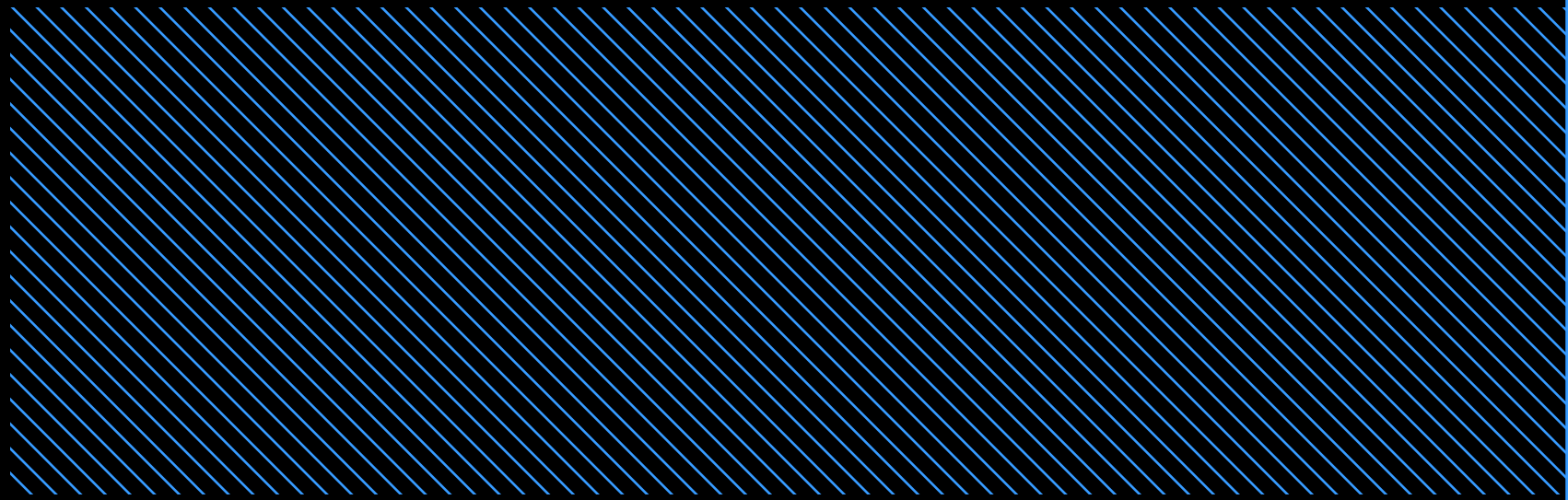
- docker-compose.yaml




```yaml
version: "3.7"

services:
  db:
    container_name: metachron-db
    image: mysql:latest
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD:
      MYSQL_DATABASE: metachron_db
      MYSQL_USER: user
      MYSQL_PASSWORD:
    ports:
      - 10000:3306
    volumes:
      - .data:/var/lib/mysql

  phpmyadmin:
    container_name: metachron-pma
    image: phpmyadmin/phpmyadmin:latest
    restart: always
    depends_on:
      - db
    links:
      - db:db
    environment:
      PMA_HOST: db
      PMA_USER: root
      PMA_PASSWORD:
    ports:
      - 11000:80

volumes:
  data:
```

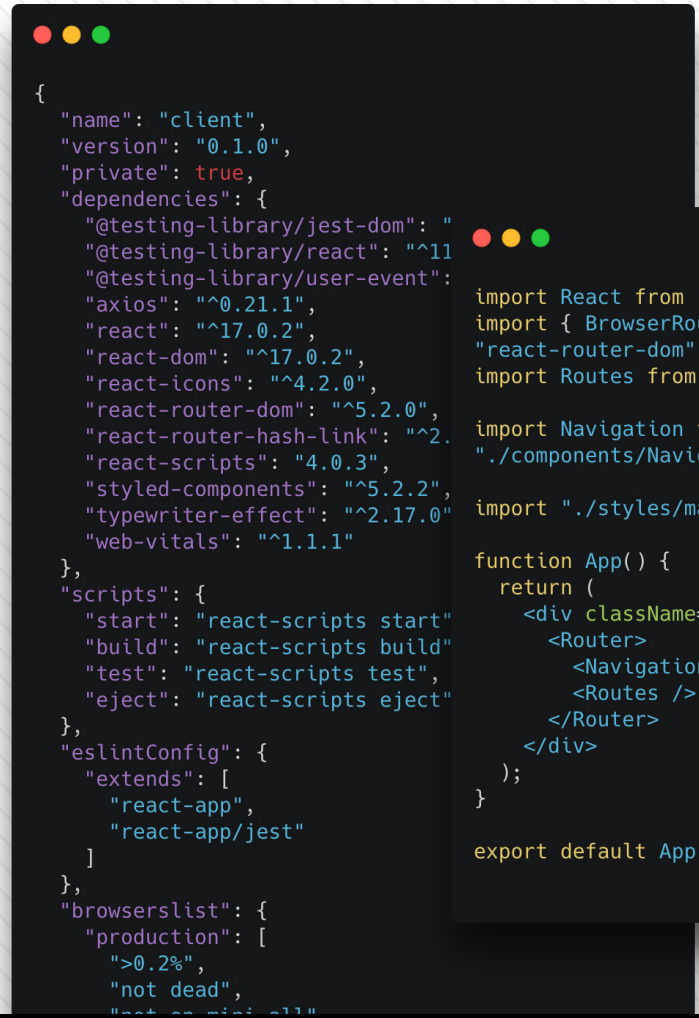
./SRC/PROJECT

./SRC/PROJECT/CLIENT



CLIENT-APP

- npm
- creating react-app for frontend
- some usefull cli-commands
    - npx create-react-app .
        - npm start
        - npm i react-router-dom
        - npm i react-icons\
        - npm i styled-components
        - npm i react-router-hash-link
        - npm i typewriter-effect
        - npm i axios



```
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "
    "@testing-library/react": "^11
    "@testing-library/user-event":
    "axios": "^0.21.1",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-icons": "^4.2.0",
    "react-router-dom": "^5.2.0",
    "react-router-hash-link": "^2.
    "react-scripts": "4.0.3",
    "styled-components": "^5.2.2",
    "typewriter-effect": "^2.17.0"
    "web-vitals": "^1.1.1"
  },
  "scripts": {
    "start": "react-scripts start"
    "build": "react-scripts build"
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
```

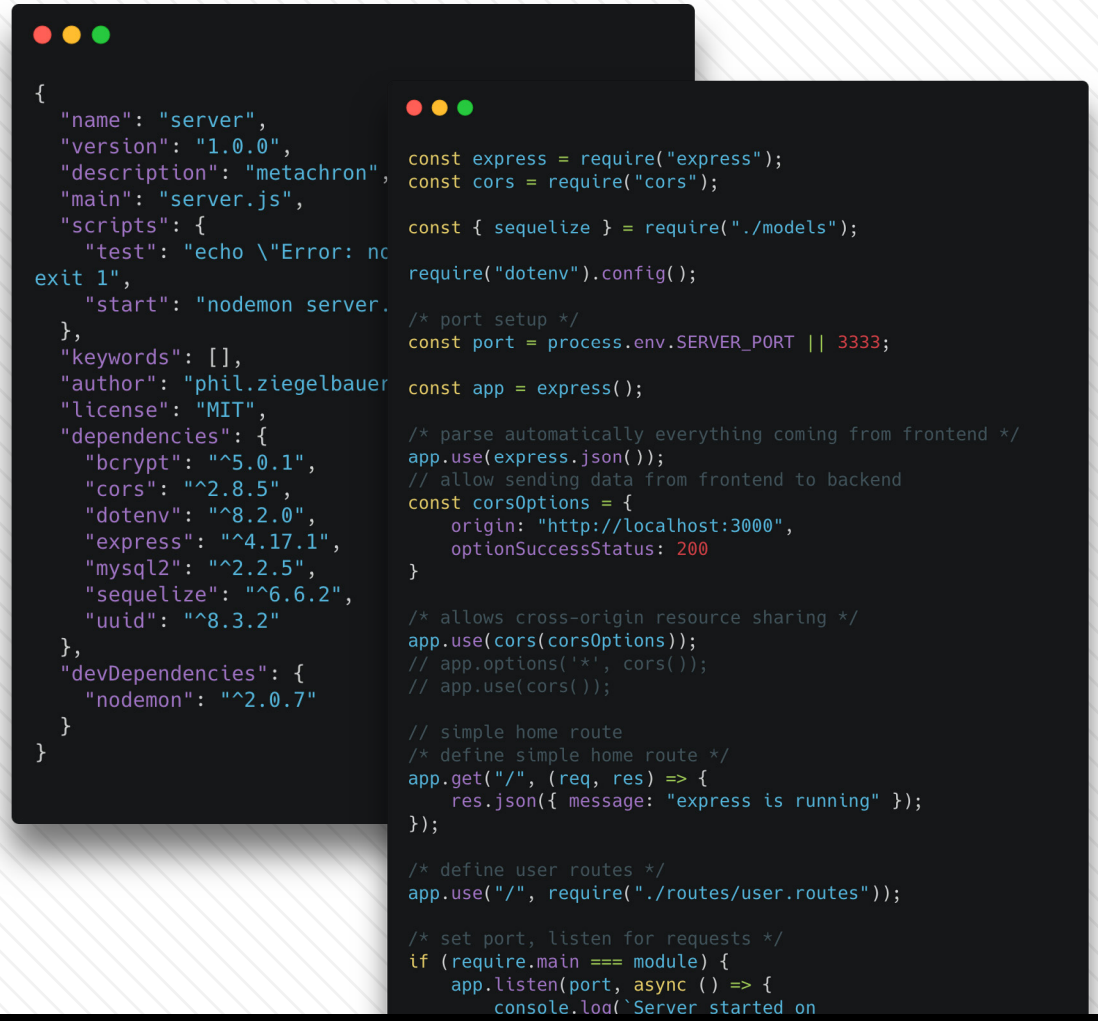
```
import React from "react";
import { BrowserRouter as Router } from
"react-router-dom";
import Routes from "./routes/Routes";

import Navigation from
"./components/Navigation/Navigation";

import "./styles/main/main-style.css";

function App() {
  return (
    <div className="App">
      <Router>
        <Navigation />
        <Routes />
      </Router>
    </div>
  );
}

export default App;
```

./SRC/PROJECT

./SRC/PROJECT/SERVER

SERVER-APP

- npm
- creating node-app for backend
- some usefull cli-commands
    - npm init .
    - npm i express
    - npm i sequelize
    - npm i mysql2
    - npm i cors
    - npm i dotenv
    - npm nodemon --save-dev

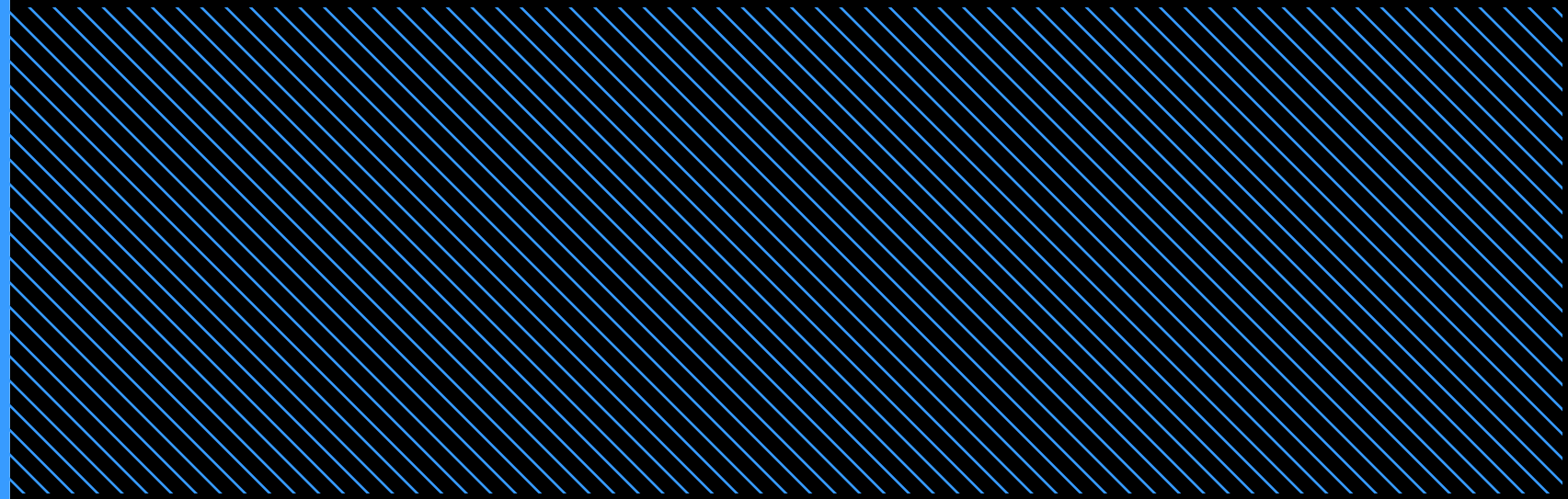


```
{
  "name": "server",
  "version": "1.0.0",
  "description": "metachron",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no
exit 1",
    "start": "nodemon server.
  },
  "keywords": [],
  "author": "phil.ziegelbauer
  "license": "MIT",
  "dependencies": {
    "bcrypt": "^5.0.1",
    "cors": "^2.8.5",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "mysql2": "^2.2.5",
    "sequelize": "^6.6.2",
    "uuid": "^8.3.2"
  },
  "devDependencies": {
    "nodemon": "^2.0.7"
  }
}
```

```
const express = require("express");
const cors = require("cors");

const { sequelize } = require("./models");

require("dotenv").config();

/* port setup */
const port = process.env.SERVER_PORT || 3333;

const app = express();

/* parse automatically everything coming from frontend */
app.use(express.json());
// allow sending data from frontend to backend
const corsOptions = {
    origin: "http://localhost:3000",
    optionSuccessStatus: 200
}

/* allows cross-origin resource sharing */
app.use(cors(corsOptions));
// app.options('*', cors());
// app.use(cors());

// simple home route
/* define simple home route */
app.get("/", (req, res) => {
    res.json({ message: "express is running" });
});

/* define user routes */
app.use("/", require("./routes/user.routes"));

/* set port, listen for requests */
if (require.main === module) {
    app.listen(port, async () => {
        console.log(`Server started on
```

./SRC/PROJECT

./SRC/PROJECT/ORM



SEQUELIZE

- some usefull cli-commands
  - sequelize db:create
  - sequelize model:generate
  - sequelize db:drop
  - sequelize db:create
  - sequelize db:migrate (:undo(:all))
  - sequelize seed:generate
  - sequelize db:seed:all
  - sequelize db:seed --seed name
  - sequelize seed:undo (:all)
  - sequelize db:migrate:status

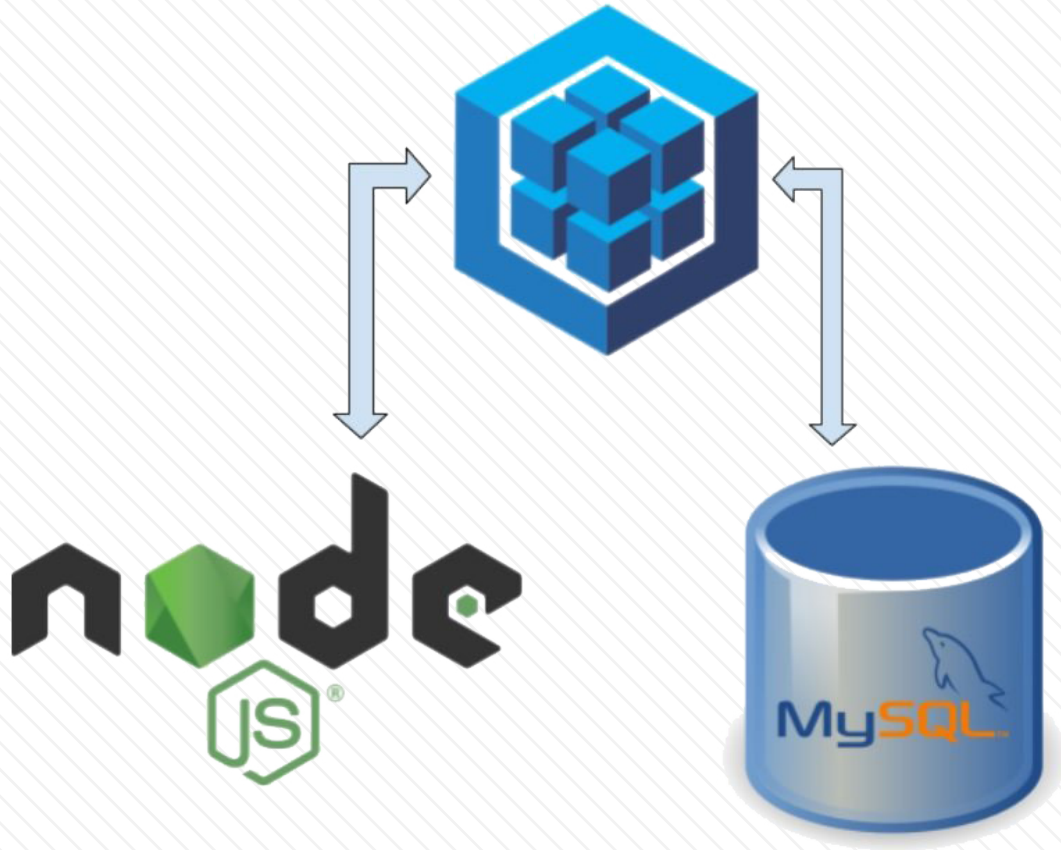
```
'use strict';

/* import bcrypt */
const bcrypt = require('bcrypt');

const {
  Model
} = require('sequelize');

/* export model definition User */
module.exports = (sequelize, DataTypes) => {
  class User extends Model {
    /**
     * Helper method for defining associations.
     * This method is not a part of Sequelize lifec
     * The `models/index` file will call this metho
     */
    static associate({ Userrole, Userdatainfo, Qual
      // define association here

      this.belongsTo(Userrole, { foreignKey: "userr

      this.belongsTo(Userdatainfo, { foreignKey: "u

      this.belongsToMany(TrainingGroup, { through:

      this.belongsToMany(Qualification, { through:

      this.hasMany(Event, { foreignKey: "trainer_id

      this.belongsToMany(Event, { through: "user_at

    }
    toJSON() {
      return { ...this.get(), id: undefined };
    }
  }

  User.init({
```

```
User.init({
  id: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true,
    allowNull: false
  },
  uuid: {
    type: DataTypes.UUID,
    defaultValue: DataTypes.UUIDV4,
    unique: true,
    allowNull: false
  },
  username: {
    type: DataTypes.STRING,
    unique: true,
    allowNull: false
  },
  password: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  pic_path: {
    type: DataTypes.STRING,
    allowNull: false
  },
  userrole_id: {
    type: DataTypes.INTEGER,
    allowNull: false
  },
  userdata_id: {
    type: DataTypes.INTEGER,
    allowNull: false
  },
  createdAt: {
    type: DataTypes.DATE,
    allowNull: false
```

NODEJS-SEQUELIZE-MYSQL

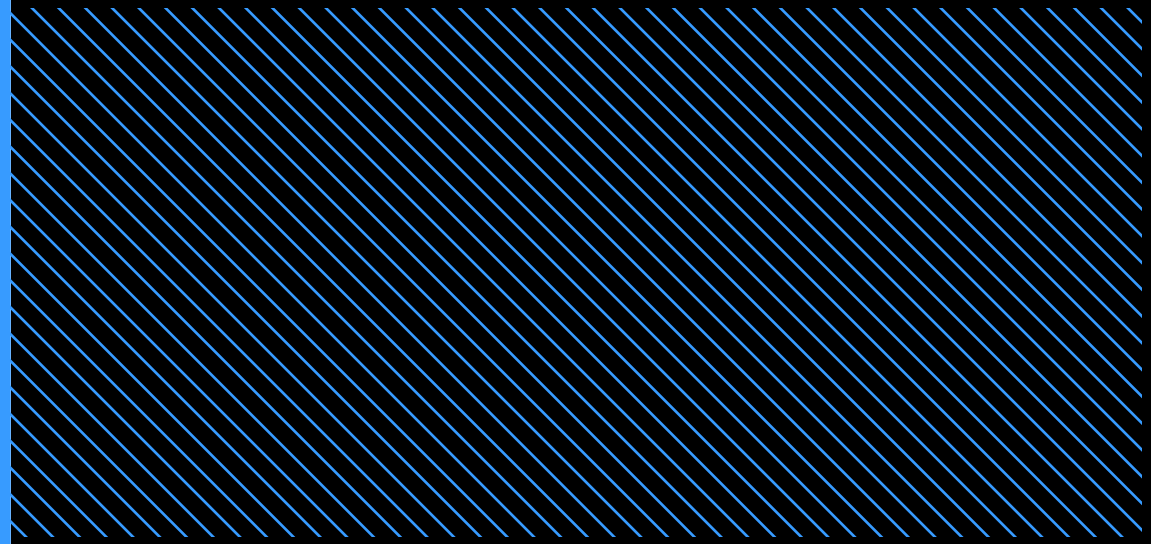
- sequelize is translation layer between nodejs and the db

- define model objects
- define model associations
- no raw query strings (select * from...)
- migrate to db
- possibility to setup seeders
- possibility to nest objects
- single language (js in our case)



./SRC/PROJECT

./SRC/PROJECT/BCRYPT



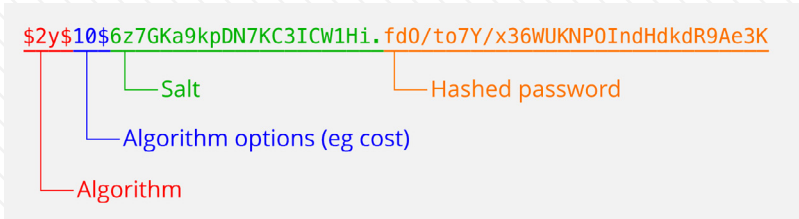
| PHIL ZIEGELBAUER

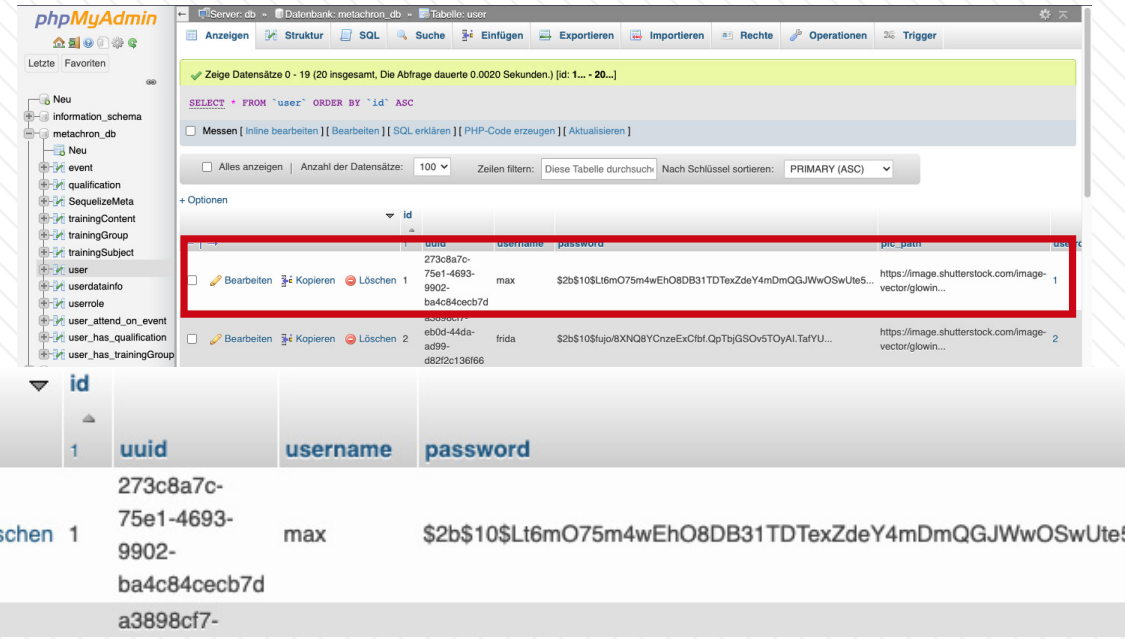
CODERS.BAY, 17.06.2021, Linz Austria |

SECURE INFORMATION

node.bcrypt.js

- bcrypt to hash information to db
(for example password)
(in our case saltrounds = 10)

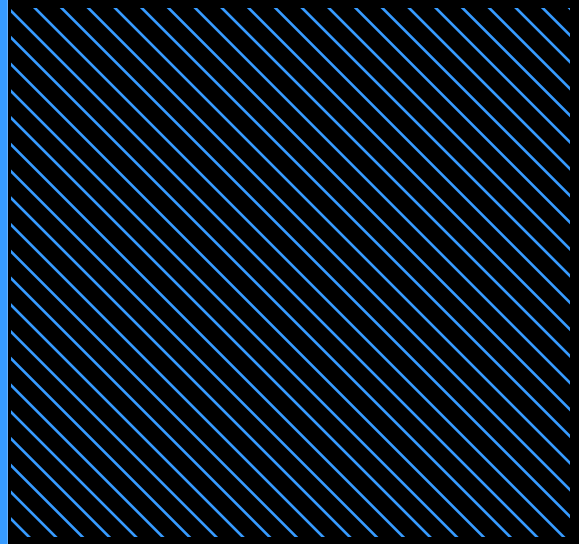




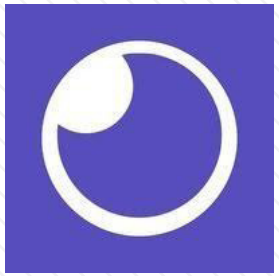
id	uuid	username	password
1	273c8a7c-75e1-4693-9902-ba4c84cecb7d	max	$2b$10$Lt6mO75m4wEhO8DB31TDTexZdeY4mDmQGJWwOSwUte5..
	a3898cf7-		

./SRC/PROJECT

./SRC/PROJECT/INSOMNIA-API-CLIENT



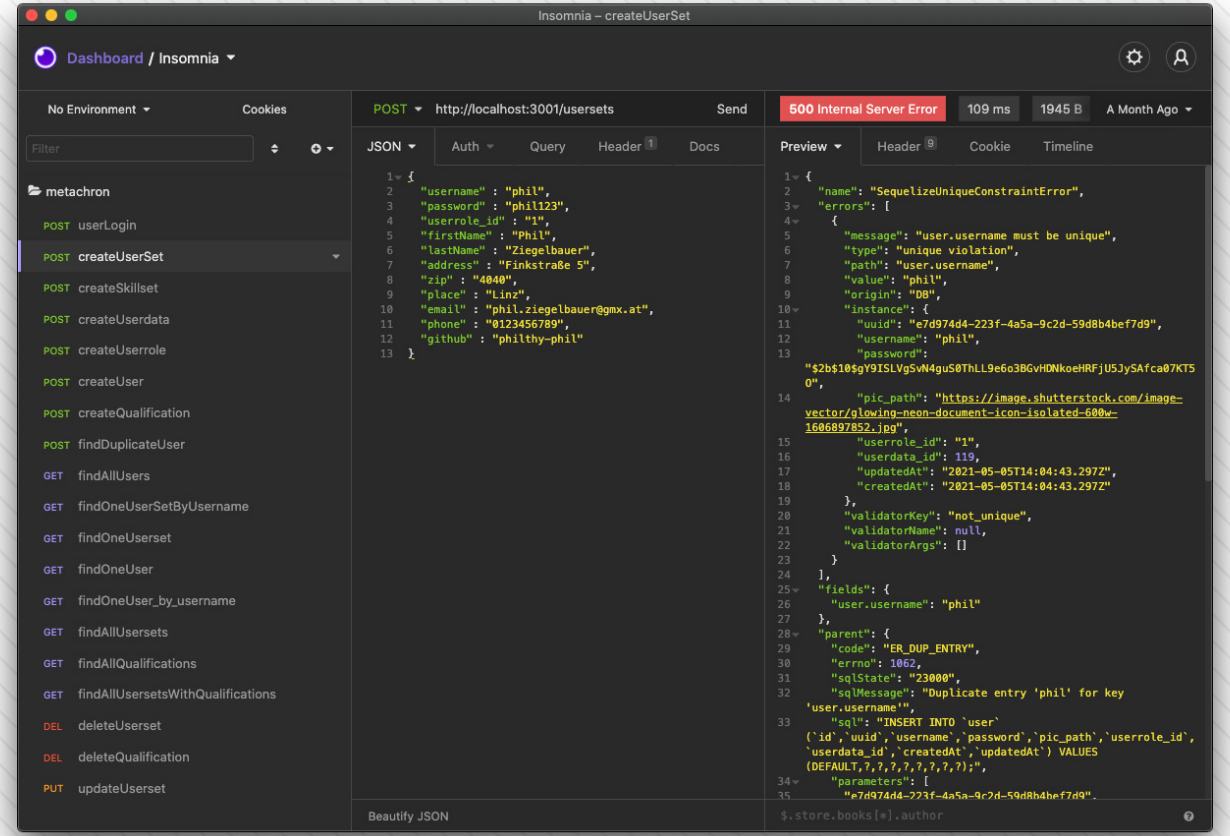
SECURE INFORMATION



- insomnia is a api-client
  (like postman just for free)

- testing our api routes (backend)

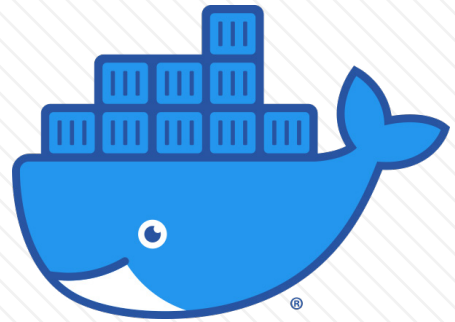
- CRUD routes

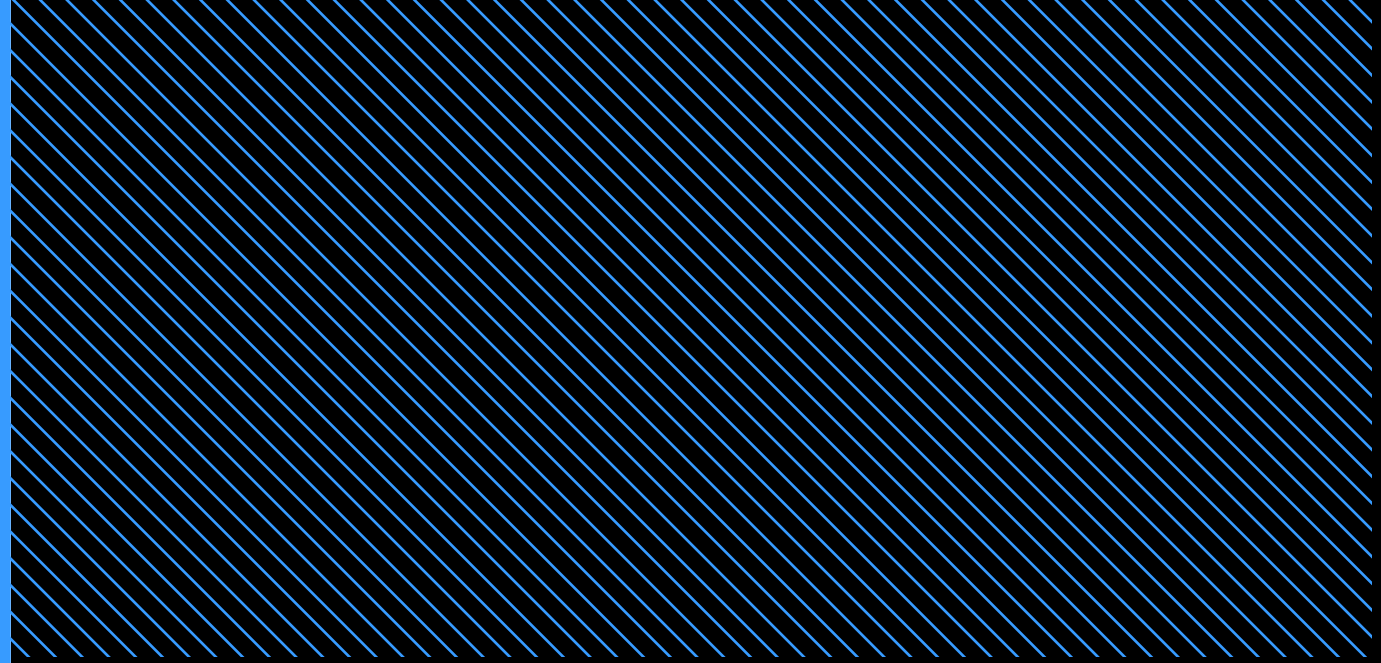


./SRC/PROJECT

./SRC/PROJECT/APPS

What the ... ?





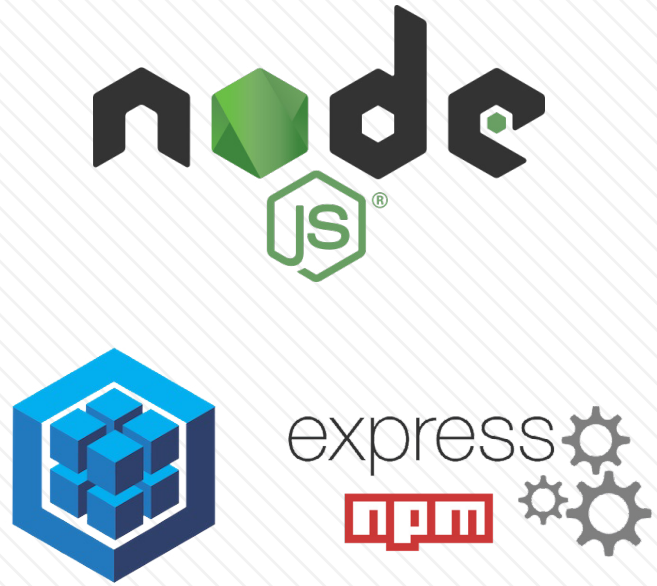
./SRC/PROJECT

./SRC/PROJECT/TEC

| PHIL ZIEGELBAUER

CODERS.BAY, 17.06.2021, Linz Austria |

How the ... ?



Versioning?

./SRC/PROJECT

./SRC/PROJECT/CODE

```
if (questions.length > 0) {
    questions.forEach (question => {
        question.answer();
    });
} else {
    return phil.thanks();
}
```