Criando a estrutura

1. Yarn init -y
2. Yarn add express
3. Yarn add nodemon -D
4. Alterar o package.jason

```
  "scripts": {
    "dev": "nodemon ./src/server.js"
  }
```

   a.

5. Criar o serve.js dentro da pasta src

```
const express = require('express')
const app = express();
app.listen(3001, ()=>{
   console.log("Servidor rodando na porta
3001")
})
```

6. Criar uma rota

```
const express = require('express')
const app = express();
app.get('/', (req, res) => {
    res.send('Olá');
});
app.listen(3001, ()=>{
    console.log("Servidor rodando na port
3001")
})
```

7. Banco de dados
   a. Node + mysql
   b. Node + query builder
   c. Node + sequelizer
8. Yarn add sequelize mysql2
9. Yarn add sequelize-cli -D
10. Yarn sequelize –help
11. Criar uma rquivos de rotas – routes.js

```
const express = require('express')
const routes = express.Router();
routes.get('/', (req,res) =>{
    return res.json({hello: "Olá"})
```

```
})
module.exports = routes;
```

12. Alterar o arquivo server.js

```
const express = require('express')
const routes = require('./routes')
const app = express();
app.use(express.json());
app.use(routes);
app.listen(3001, ()=>{
    console.log("Servidor rodando na porta
3001")
})
```

13. Yarn dev
14. Configurar as credenciais de acesso a base de dados
    a. Criar uma pasta src/config
    b. Criar um arquivo database.js

```
module.exports = {
    dialect: 'mysql',
    host: 'localhost',
    username: 'root',
    password:'123456',
    database: 'sqlNode',
    define : {
        timestamps: true,
        underscored: true
    }
}
```

    c. Criar uma pasat src/database/index.js

```
const Sequelize = require('sequelize')
const dbConfig = require('../config/da
tabase')

const connection = new Sequelize(dbCon
fig);
```

```
module.exports = connection;
```

d. criar o arquivo na raiz .sequelizerc

```
const path = require('path');

module.exports = {
    config: path.resolve(__dirname,'src','config','database.js'),
}
```

e. yarn sequelize db:create

15. Criar a primeira tabela
    a. Alterar o arquivo .sequelizerc

```
const path = require('path');

module.exports = {
    config: path.resolve(__dirname,'src','config','database.js'),
    'migrations-path': path.resolve(__dirname, 'src', 'database', 'migrations')
}
```

    b. Criar a pasta src/database/migrations
    c. Yarn sequelize migration:create --name=create-users
    d. Criou o arquivo migrations

```
'use strict';

module.exports = {
  up: (queryInterface, Sequelize) => {
// dizer o que a migrations vai realizar na bd
    /*
      Add altering commands here.
      Return a promise to correctly handle asynchronicity.

      Example:
```

```
      return queryInterface.createTabl
e('users', { id: Sequelize.INTEGER });
    */
  },

  down: (queryInterface, Sequelize) =>
{ // se der alguma coisa errada o que
tenho que fazer
    /*
      Add reverting commands here.
      Return a promise to correctly ha
ndle asynchronicity.

      Example:
      return queryInterface.dropTable(
'users');
    */
  }
};
```

e. Alterar o migrations

```
'use strict';
module.exports = {
  up: (queryInterface, Sequelize) => { // dizer
o que a migrations vai realizar na bd
    return queryInterface.createTable('users', {
      id: {
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true,
        allowNull: false,
      },
      name: {
        type: Sequelize.STRING,
        allowNull: false,
```

```
      },
      email: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      created_at:{
        type: Sequelize.DATE,
        allowNull: false,
      },
      updated_at:{
        type: Sequelize.DATE,
        allowNull: false,
      }
    });

  },
  down: (queryInterface, Sequelize) => { // se d
er alguma coisa errada o que tenho que fazer
    return queryInterface.dropTable('users');

  }
};
```

f. Yarn sequelize db:migrate

g. Para voltar yarn sequelize db:migrate:undo

16. Criar a pasta model dentro do src

a. Criar o users.js

```
const {Model, DataTypes} = require('se
quelize');

class User extends Model{
    static init(sequelize){
        super.init({
            name: DataTypes.STRING,
            email: DataTypes.STRING,
        },  {
```

```
            sequelize
        })
    }
}

module.exports = User;
```

b. Arquivo index.js da pasta database

```
const Sequelize = require('sequelize')
const dbConfig = require('../config/da
tabase')

const User = require('../models/users'
)

const connection = new Sequelize(dbCon
fig);

User.init(connection);

module.exports = connection;
```

17. Alterar o server.js

```
const express = require('express')
const routes = require('./routes')

require('./database')

const app = express();

app.use(express.json());
app.use(routes);

app.listen(3001, ()=>{
    console.log("Servidor rodando na p
orta 3001")
})
```

18. Criar uma pasta src/controllers

```
const User = require('../models/users'
)

module.exports = {
    async store (req,res) {
        const {name, email} = req.body
;

        const user = await User.create
({name, email});

        return res.json(user)
    }

};
```

b. Alterar o arquivo de rotas

```
const express = require('express')
const UserController = require('./cont
rollers/UserController')

const routes = express.Router();

routes.get('/', (req,res) =>{
    return res.json({hello: "Olá"})
})

routes.post('/users', UserController.s
tore);

module.exports = routes;
```

19. Criar o método de listar todos no BD

a. Alterar o userController criando o método indexAll

```
const User = require('../models/User')
```

```
module.exports = {
    async store (req,res) {
        const {name, email} = req.body;
        console.log(name, email)
        const user = await User.create({name, em
ail});
        return res.json(user)
    },
    async indexAll(req,res){
        const users = await User.findAll();
        return res.json(users);
    }

};
```

b. Adicionar no arquivo Rotas

```
routes.get('/users', UserController.indexAll);
```

c. Testar o get no insomnia

20. Relacionamentos
    a. Usuario pode ter vários endereços no banco de dados
    b. yarn sequelize migration:create --name=create-addresses
    c. criar a migration endereço

```
'use strict';
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('addresses
', {
      id: {
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true,
        allowNull: false,
      },
      user_id:{
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {model: 'users', key: 'id'},
```

```
        onUpdate: 'CASCADE',
        onDelete: 'CASCADE',
      },
      CEP: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      rua: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      numero: {
        type: Sequelize.INTEGER,
        allowNull: false,
      },
      created_at:{
        type: Sequelize.DATE,
        allowNull: false,
      },
      updated_at:{
        type: Sequelize.DATE,
        allowNull: false,
      }
    });

  },
  down: (queryInterface, Sequelize) => {
    return queryInterface.dropTable('addresses')
;
 }
};
```

d. Yarn sequelize db:migrate

e. Criar o model

```
const {Model, DataTypes} = require('sequelize');
class Adsress extends Model{
```

```
    static init(sequelize){
        super.init({
            CEP: DataTypes.STRING,
            rua: DataTypes.STRING,
            numero: DataTypes.INTEGER,
        },  {
            sequelize
        })
    }
   static associate(models){
       this.belongsTo(models.User, {foreignKey :
'user_id', as : 'possuiUsuario'}) //endereço
possui um dono
    }

 }
 module.exports = Address;
```

f. Importar o model dentro do arquivo index do banco de dados
   i. Migrations/index

```
const Sequelize = require('sequelize')
const dbConfig = require('../config/database')
const User = require('../models/User')
const Address = require('../models/Address')
const connection = new Sequelize(dbConfig);
User.init(connection);
Address.init(connection)
Address.associate(connection.models)
module.exports = connection;
```

g. Alterar a rota de adicionar endereço

```
const AddressController = require('./controllers
/AddressController')
routes.post('/users/:user_id/address', AddressContro
ller.store);
```

h. Criar o AddressController

```
const Address = require('../models/Address')
```

```
const User = require('../models/User')

module.exports = {
    async store(req, res) {
        const { CEP, rua, numero } = req.body;
        console.log(CEP, rua, numero)
        const { user_id } = req.params;
        //verificar se o usuario existe
        const user = await User.findByPk(user_id
)
        if (!user) {
            return res.status(400).json({ error:
'Usuario não encontrado' })
        }

        const address = await Address.create({
            CEP,
            rua,
            numero,
            user_id
        });

        return res.json(address)
    },
};
```

21. Buscar endereço de acrodo com o usuário
    a. Criar uma rota do tipo get

```
routes.get('/users/:user_id/addresses', AddressC
ontroller.index);
```

    b. Realizar a associação de um usuário possu vários endereços

```
const {Model, DataTypes} = require('sequelize');

class User extends Model{
    static init(sequelize){
        super.init({
```

```
            name: DataTypes.STRING,
            email: DataTypes.STRING,
        },  {
            sequelize
        })
    }
    static associate(models){
        this.hasMany(models.Address, {foreignKey
: 'user_id', as : 'enderecos'})
    }
}

module.exports = User;
```

c. Acrescentar o método associate no index.js

```
User.associate(connection.models)
```

d. Alterar o AddressControllers

```
const Address = require('../models/Address')
const User = require('../models/User')

module.exports = {

    async index(res,req){
        const { user_id} = req.params;
        const user = await User.findByPk(user_id
,{
            include: { association: 'enderecos'}
        })


        //const addresses = await Address.findAl
l({where: {user_id} })
        //return res.json(addresses)

        return res.json(user)
    },
```

```
    async store(req, res) {
        const { cep, rua, numero } = req.body;
        console.log(cep, rua, numero)
        const { user_id } = req.params;
        //verificar se o usuario existe
        const user = await User.findByPk(user_id
)
        if (!user) {
            return res.status(400).json({ error:
'Usuario não encontrado' })
        }

        const address = await Address.create({
            cep,
            rua,
            numero,
            user_id
        });

        return res.json(address)
    },
};
```

xxxix. Alterar

22. Criar tabela N-N
    a. yarn sequelie migration:create --name=create-techs
    b. editar o migration techs

```
'use strict';

module.exports = {
  up: (queryInterface, Sequelize) => {
      return queryInterface.createTable('techs',
{
        id: {
          type: Sequelize.INTEGER,
```

```
          primaryKey: true,
          autoIncrement: true,
          allowNull: false,
        },
        name:{
          type: Sequelize.STRING,
          allowNull: false,
        },
        created_at:{
          type: Sequelize.DATE,
          allowNull: false,
        },
        updated_at:{
          type: Sequelize.DATE,
          allowNull: false,
        },
      });

  },

  down: (queryInterface, Sequelize) => {

      return queryInterface.dropTable('users');

  }
};
```

c. yarn sequelie migration:create --name=create-user_techs

d. Alterar o migration user_techs

```
'use strict';

module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('user_tech
s', {
      id: {
```

```
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true,
        allowNull: false,
      },
      user_id:{
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {model: 'users', key: 'id'},
        onUpdate: 'CASCADE',
        onDelete: 'CASCADE',
      },
      tech_id:{
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {model: 'techs', key: 'id'},
        onUpdate: 'CASCADE',
        onDelete: 'CASCADE',
      },
      created_at: {
        type: Sequelize.DATE,
        allowNull: false,
      },
      updated_at: {
        type: Sequelize.DATE,
        allowNull: false,
      },
    });

 },

  down: (queryInterface, Sequelize) => {
    return queryInterface.dropTable('user_techs'
);
```

```
  }
};
```

e. yarn sequelize db:migrate

f. Criar um model techs

```
const {Model, DataTypes} = require('sequelize');

class Tech extends Model{
    static init(sequelize){
        super.init({
            name: DataTypes.STRING,
        },  {
            sequelize,
            tableName: 'techs'
        })
    }
    static associate(models){
        this.belongsToMany(models.User, {
            foreignKey : 'tech_id',
            through: 'user_techs',
            as : 'users'})
    }

}

module.exports = Tech;
```

g. Alterar o models Usuario

```
const {Model, DataTypes} = require('se
quelize');

class User extends Model{
    static init(sequelize){
        super.init({
            name: DataTypes.STRING,
            email: DataTypes.STRING,
```

```
        }, {
            sequelize
        })
    }
    static associate(models){
        this.hasMany(models.Address, {
foreignKey : 'user_id', as : 'endereco
s'})
        this.belongsToMany(models.Tech
, {
            foreignKey : 'user_id',
            through: 'user_techs',
            as : 'techs'})
        }
    }

module.exports = User;
```

h. Inicializar o model de tech no arquivo index.js

```
const Sequelize = require('sequelize')
const dbConfig = require('../config/database')

const User = require('../models/User')
const Address = require('../models/Address')
const Tech = require('../models/Tech')

const connection = new Sequelize(dbConfig);

User.init(connection);
Address.init(connection)
Tech.init(connection)

Address.associate(connection.models)
User.associate(connection.models)
Tech.associate(connection.models)
```

```
module.exports = connection;
```

i. Criar as rotas de get e store

```
const TechController = require('./controllers/TechController')
routes.post('/users/:user_id/techs', TechController.store);
routes.get('/users/:user_id/techs', TechController.index);
```

j. controlerTech – store

```
const Tech = require('../models/Tech')
const User = require('../models/User')

module.exports = {

    async index(req,res){
    },

    async store(req, res) {
        const {user_id} = req.params;
        const {name} = req.body;

        const user = await User.findByPk(user_id)
        if (!user) {
            return res.status(400).json({ error: 'Usuario não encontrado' })
        }
        //created bolean que define se a tecnologia foi criada ou não
        const [tech, created] = await Tech.findOrCreate({
            where : {name}
        })

        await user.addTech(tech);
```

```
        return res.json(tech)
    },
};
```

k. Deletar uma tecnologia do usuário

```
async delete(req,res){
        const {user_id} = req.params;
        const {name} = req.body;
        const user = await User.findBy
Pk(user_id)
        if (!user) {
            return res.status(400).jso
n({ error: 'Usuario não encontrado' })
        }

        const tech = await Tech.findOn
e({
            where : {name}
        })

        await user.removeTech(tech)

        return res.json({mensage : "Re
movida com sucesso"})
    }
```

l. Adicionar a rota delete

```
routes.delete('/users/:user_id/techs',
TechController.delete);
```

m. Listagem

n.