

Criando a estrutura

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

```
"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 + sequelize
8. Yarn add sequelize mysql2
9. Yarn add sequelize-cli -D
10. Yarn sequelize --help
11. Criar um arquivo 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

```
i   module.exports = {  
ii  dialect: 'mysql',  
iii host: 'localhost',  
iv  username: 'root',  
v   password: '123456',  
vi  database: 'sqlNode',  
vii define : {  
viii   timestamps: true,  
ix     underscored: true  
x     }  
xi  }
```

- c. Criar uma pasta src/database/index.js

```
i   const Sequelize = require('sequelize')  
ii  const dbConfig = require('../config/database')  
iii  
iv  const connection = new Sequelize(dbConfig);  
v
```

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

d. criar o arquivo na raiz .sequelizeerc

```
i const path = require('path');  
ii  
iii module.exports = {  
iv   config: path.resolve(__dirname, 'src', 'config', 'database.js'),  
v };
```

e. yarn sequelize db:create

15. Criar a primeira tabela

a. Alterar o arquivo .sequelizeerc

```
i const path = require('path');  
ii  
iii module.exports = {  
iv   config: path.resolve(__dirname, 'src', 'config', 'database.js'),  
v   'migrations-  
path': path.resolve(__dirname, 'src',  
'database', 'migrations')  
vi }
```

b. Criar a pasta src/database/migrations

c. Yarn sequelize migration:create --name=create-users

d. Criou o arquivo migrations

```
i 'use strict';  
ii  
iii module.exports = {  
iv   up: (queryInterface, Sequelize) => {  
// dizer o que a migrations vai realiz  
ar na bd  
v   /*  
vi       Add altering commands here.  
vii      Return a promise to correctly ha  
ndle asynchronicity.  
viii  
ix      Example:
```

```

x      return queryInterface.createTable
xi      e('users', { id: Sequelize.INTEGER });
xii     */
xiii    },
xiv     down: (queryInterface, Sequelize) =>
xv     { // se der alguma coisa errada o que
xvi     tenho que fazer
xvii     /*
xviii     Add reverting commands here.
xix      Return a promise to correctly handle
xx      asynchronicity.
xxi      Example:
xxii     return queryInterface.dropTable(
xxiii    'users');
        */
        }
    };

```

e. Alterar o migrations

```

i      'use strict';
ii     module.exports = {
iii     up: (queryInterface, Sequelize) => { // dizer
iv     o que a migrations vai realizar na bd
v      return queryInterface.createTable('users', {
vi      id: {
vii      type: Sequelize.INTEGER,
viii     primaryKey: true,
ix      autoIncrement: true,
x      allowNull: false,
xi     },
xii     name: {
xiii    type: Sequelize.STRING,
        allowNull: false,

```

```

xiv      },
xv       email: {
xvi         type: Sequelize.STRING,
xvii        allowNull: false,
xviii     },
xix      created_at:{
xx        type: Sequelize.DATE,
xxi        allowNull: false,
xxii       },
xxiii    updated_at:{
xxiv        type: Sequelize.DATE,
xxv         allowNull: false,
xxvi        }
xxvii    });
xxviii
xxix     },
xxx      down: (queryInterface, Sequelize) => { // se d
er alguma coisa errada o que tenho que fazer
xxxi      return queryInterface.dropTable('users');
xxxii
xxxiii   }
xxxiv    };

```

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

```

i      const {Model, DataTypes} = require('se
quelize');
ii
iii     class User extends Model{
iv       static init(sequelize){
v         super.init({
vi           name: DataTypes.STRING,
vii          email: DataTypes.STRING,
viii         }, {

```

```

ix         sequelize
x         })
xi       }
xii      }
xiii
xiv      module.exports = User;

```

b. Arquivo index.js da pasta database

```

i      const Sequelize = require('sequelize')
ii     const dbConfig = require('../config/database')
iii
iv     const User = require('../models/users')
v
vi     const connection = new Sequelize(dbConfig);
vii
viii   User.init(connection);
ix
x      module.exports = connection;

```

17. Alterar o server.js

```

i      const express = require('express')
ii     const routes = require('./routes')
iii
iv     require('./database')
v
vi     const app = express();
vii
viii   app.use(express.json());
ix     app.use(routes);
x
xi     app.listen(3001, ()=>{
xii       console.log("Servidor rodando na porta 3001")
xiii    })

```

18. Criar uma pasta src/controllers

```
i   const User = require('../models/users')
ii
iii  module.exports = {
iv   async store (req,res) {
v     const {name, email} = req.body
vi
vii   const user = await User.create
      ({name, email});
viii
ix     return res.json(user)
x
xi   }
xii };;
```

b. Alterar o arquivo de rotas

```
i   const express = require('express')
ii  const UserController = require('./controllers/UserController')
iii
iv  const routes = express.Router();
v
vi  routes.get('/', (req,res) =>{
vii    return res.json({hello: "Olá"})
viii })
ix
x   routes.post('/users', UserController.store);
xi
xii module.exports = routes;
```

19. Criar o método de listar todos no BD

a. Alterar o userController criando o método indexAll

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

```

ii  module.exports = {
iii    async store (req,res) {
iv      const {name, email} = req.body;
v      console.log(name, email)
vi      const user = await User.create({name, email});
vii     return res.json(user)
viii    },
ix    async indexAll(req,res){
x      const users = await User.findAll();
xi      return res.json(users);
xii    }
xiii
xiv  };

```

b. Adicionar no arquivo Rotas

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

c. Testar o get no insomnia

20. Relacionamentos

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

```

i  'use strict';
ii  module.exports = {
iii    up: (queryInterface, Sequelize) => {
iv      return queryInterface.createTable('addresses', {
v        id: {
vi          type: Sequelize.INTEGER,
vii         primaryKey: true,
viii        autoIncrement: true,
ix          allowNull: false,
x          },
xi         user_id:{
xii          type: Sequelize.INTEGER,
xiii         allowNull: false,
xiv         references: {model: 'users', key: 'id'},

```



```

xv      onUpdate: 'CASCADE',
xvi     onDelete: 'CASCADE',
xvii    },
xviii   CEP: {
xix     type: Sequelize.STRING,
xx      allowNull: false,
xxi     },
xxii    rua: {
xxiii   type: Sequelize.STRING,
xxiv    allowNull: false,
xxv     },
xxvi    numero: {
xxvii   type: Sequelize.INTEGER,
xxviii  allowNull: false,
xxix    },
xxx     created_at:{
xxxi    type: Sequelize.DATE,
xxxii   allowNull: false,
xxxiii  },
xxxiv   updated_at:{
xxxv    type: Sequelize.DATE,
xxxvi   allowNull: false,
xxxvii  }
xxxviii });
xxxix

xl      },
xli     down: (queryInterface, Sequelize) => {
xlii    return queryInterface.dropTable('addresses')
;
xliii   }
xliv    };

```

d. Yarn sequelize db:migrate

e. Criar o model

```

i      const {Model, DataTypes} = require('sequelize');
ii     class Adress extends Model{

```

```

iii     static init(sequelize){
iv       super.init({
v         CEP: DataTypes.STRING,
vi        rua: DataTypes.STRING,
vii       numero: DataTypes.INTEGER,
viii      }, {
ix        sequelize
x         })
xi       }

static associate(models){

    this.belongsTo(models.User, {foreignKey :
'user_id', as : 'possuiUsuario'}) //endereço
possui um dono

    }

xii
xiii   }
xiv    module.exports = Address;

```

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

```

ii     const Sequelize = require('sequelize')
iii    const dbConfig = require('../config/database')
iv     const User = require('../models/User')
v      const Address = require('../models/Address')
vi     const connection = new Sequelize(dbConfig);
vii    User.init(connection);
viii   Address.init(connection)
ix     Address.associate(connection.models)
x      module.exports = connection;

```

- g. Alterar a rota de adicionar endereço

```

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

```

- h. Criar o AddressController

```

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

```

```

ii   const User = require('../models/User')
iii
iv   module.exports = {
v     async store(req, res) {
vi       const { CEP, rua, numero } = req.body;
vii      console.log(CEP, rua, numero)
viii     const { user_id } = req.params;
ix       //verificar se o usuario existe
x       const user = await User.findByPk(user_id
xi     )
xii      if (!user) {
xiii        return res.status(400).json({ error:
xiv      'Usuario não encontrado' })
xv      }
xvi      const address = await Address.create({
xvii        CEP,
xviii       rua,
xix        numero,
xx         user_id
xxi      });
xxii     return res.json(address)
xxiii    },
xxiv   };

```

21. Buscar endereço de acordo com o usuário

a. Criar uma rota do tipo get

```

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

```

b. Realizar a associação de um usuário possu vários endereços

```

i   const {Model, DataTypes} = require('sequelize');
ii
iii  class User extends Model{
iv    static init(sequelize){
v      super.init({

```

```

vi         name: DataTypes.STRING,
vii        email: DataTypes.STRING,
viii       }, {
ix         sequelize
x          })
xi         }
xii        static associate(models){
xiii       this.hasMany(models.Address, {foreignKey
xiv        : 'user_id', as : 'enderecos'})
xv         }
xvi        }
xvii       module.exports = User;

```

c. Acrescentar o método associate no index.js

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

d. Alterar o AddressControllers

```

i   const Address = require('../models/Address')
ii  const User = require('../models/User')
iii
iv  module.exports = {
v
vi      async index(res, req){
vii      const { user_id } = req.params;
viii     const user = await User.findByPk(user_id
ix      ,{
x        include: { association: 'enderecos' }
xi      })
xii
xiii     //const addresses = await Address.findAll(
xiv     l({where: {user_id} })
xv     //return res.json(addresses)
xvi
xvii    return res.json(user)

```

```

xviii
xix      async store(req, res) {
xx        const { cep, rua, numero } = req.body;
xxi       console.log(cep, rua, numero)
xxii      const { user_id } = req.params;
xxiii     //verificar se o usuario existe
xxiv      const user = await User.findByPk(user_id
xxv      )
xxvi      if (!user) {
xxvii         return res.status(400).json({ error:
xxviii         'Usuario não encontrado' })
xxix      }
xxx       const address = await Address.create({
xxxi         cep,
xxxii        rua,
xxxiii       numero,
xxxiv        user_id
xxxv      });
xxxvi      return res.json(address)
xxxvii    },
xxxviii  };

```

xxxix. Alterar

22. Criar tabela N-N

- yarn sequelize migration:create --name=create-techs
- editar o migration techs

```

i      'use strict';
ii
iii     module.exports = {
iv       up: (queryInterface, Sequelize) => {
v         return queryInterface.createTable('techs',
vi         {
vii          id: {
              type: Sequelize.INTEGER,

```

```

viii      primaryKey: true,
ix         autoIncrement: true,
x          allowNull: false,
xi         },
xii        name:{
xiii       type: Sequelize.STRING,
xiv        allowNull: false,
xv         },
xvi        created_at:{
xvii       type: Sequelize.DATE,
xviii      allowNull: false,
xix        },
xx         updated_at:{
xxi        type: Sequelize.DATE,
xxii       allowNull: false,
xxiii      },
xxiv       });
xxv
xxvi      },
xxvii
xxviii     down: (queryInterface, Sequelize) => {
xxix
xxx        return queryInterface.dropTable('users');
xxxi
xxxii     }
xxxiii    };

```

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

d. Alterar o migration user_techs

```

i      'use strict';
ii
iii     module.exports = {
iv       up: (queryInterface, Sequelize) => {
v         return queryInterface.createTable('user_tech
vi        s', {
          id: {

```

```
vii      type: Sequelize.INTEGER,
viii     primaryKey: true,
ix       autoIncrement: true,
x        allowNull: false,
xi       },
xii      user_id:{
xiii     type: Sequelize.INTEGER,
xiv      allowNull: false,
xv       references: {model: 'users', key: 'id'},
xvi      onUpdate: 'CASCADE',
xvii     onDelete: 'CASCADE',
xviii    },
xix      tech_id:{
xx       type: Sequelize.INTEGER,
xxi      allowNull: false,
xxii     references: {model: 'techs', key: 'id'},
xxiii    onUpdate: 'CASCADE',
xxiv     onDelete: 'CASCADE',
xxv      },
xxvi     created_at: {
xxvii    type: Sequelize.DATE,
xxviii   allowNull: false,
xxix     },
xxx      updated_at: {
xxxi     type: Sequelize.DATE,
xxxii    allowNull: false,
xxxiii   },
xxxiv    });
xxxv
xxxvi    },
xxxvii
xxxviii  down: (queryInterface, Sequelize) => {
xxxix    return queryInterface.dropTable('user_techs'
);
```

xl
xli }
xlii };

e. yarn sequelize db:migrate

f. Criar um model techs

i const {Model, DataTypes} = require('sequelize');
ii
iii class Tech extends Model{
iv static init(sequelize){
v super.init({
vi name: DataTypes.STRING,
vii }, {
viii sequelize,
ix tableName: 'techs'
x })
xi }
xii static associate(models){
xiii this.belongsToMany(models.User, {
xiv foreignKey : 'tech_id',
xv through: 'user_techs',
xvi as : 'users'})
xvii }
xviii
xix }
xx
xxi module.exports = Tech;

g. Alterar o models Usuario

i const {Model, DataTypes} = require('sequelize');
ii
iii class User extends Model{
iv static init(sequelize){
v super.init({
vi name: DataTypes.STRING,
vii email: DataTypes.STRING,


```

viii      }, {
ix          sequelize
x          })
xi      }
xii      static associate(models){
xiii          this.hasMany(models.Address, {
foreignKey : 'user_id', as : 'enderecos'})
xiv          this.belongsToMany(models.Tech
, {
xv              foreignKey : 'user_id',
xvi              through: 'user_techs',
xvii             as : 'techs'})
xviii      }
xix      }
xx
xxi      module.exports = User;

```

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

```

i      const Sequelize = require('sequelize')
ii     const dbConfig = require('../config/database')
iii
iv     const User = require('../models/User')
v     const Address = require('../models/Address')
vi     const Tech = require('../models/Tech')
vii
viii    const connection = new Sequelize(dbConfig);
ix
x      User.init(connection);
xi     Address.init(connection)
xii    Tech.init(connection)
xiii
xiv    Address.associate(connection.models)
xv    User.associate(connection.models)
xvi    Tech.associate(connection.models)

```

xvii `module.exports = connection;`

i. Criar as rotas de get e store

```
i   const TechController = require('./cont
ii  routes.post('/users/:user_id/techs', T
echController.store);
iii routes.get('/users/:user_id/techs', Te
chController.index);
```

j. controlerTech – store

```
i   const Tech = require('../models/Tech')
ii  const User = require('../models/User')
iii
iv  module.exports = {
v
vi    async index(req, res){
vii  },
viii
ix    async store(req, res) {
x      const {user_id} = req.params;
xi      const {name} = req.body;
xii
xiii     const user = await User.findBy
Pk(user_id)
xiv     if (!user) {
xv       return res.status(400).jso
n({ error: 'Usuario não encontrado' })
xvi     }
xvii     //created boolean que define se
a tecnologia foi criada ou não
xviii     const [tech, created] = await
Tech.findOrCreate({
xix       where : {name}
xx      })
xxi
xxii     await user.addTech(tech);
```

```
xxiii  
xxiv     return res.json(tech)  
xxv     },  
xxvi };
```

k. Deletar uma tecnologia do usuário

```
i   async delete(req,res){  
ii     const {user_id} = req.params;  
iii    const {name} = req.body;  
iv     const user = await User.findBy  
Pk(user_id)  
v     if (!user) {  
vi       return res.status(400).json({ error: 'Usuario não encontrado' })  
vii    }  
viii  
ix     const tech = await Tech.findOne(  
x       where : {name}  
xi     })  
xii  
xiii    await user.removeTech(tech)  
xiv  
xv     return res.json({message : "Remoção com sucesso"})  
xvi  }
```

l. Adicionar a rota delete

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

m. Listagem

n.