ساختار بندی پروژه بکند

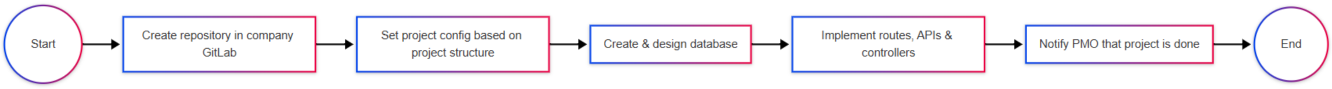
در این بخش مرحله به مرحله توسعه بکند پروژه را شرح میدهم.

- مراحل توسعه بکاند
- ساخت Repository
- ساختار بندی پروژه
- طراحی ساختار دیتابیس (ER-Diagram)
- نوشتن APi, Route , Controller
- نکات مهم پایانی

مراحل توسعه بکاند

در این قسمت بهصورت کامل مراحل توسعه بکاند با ساختار پروژه مورد نظر شرکت ماورا توضیح داده میشود.

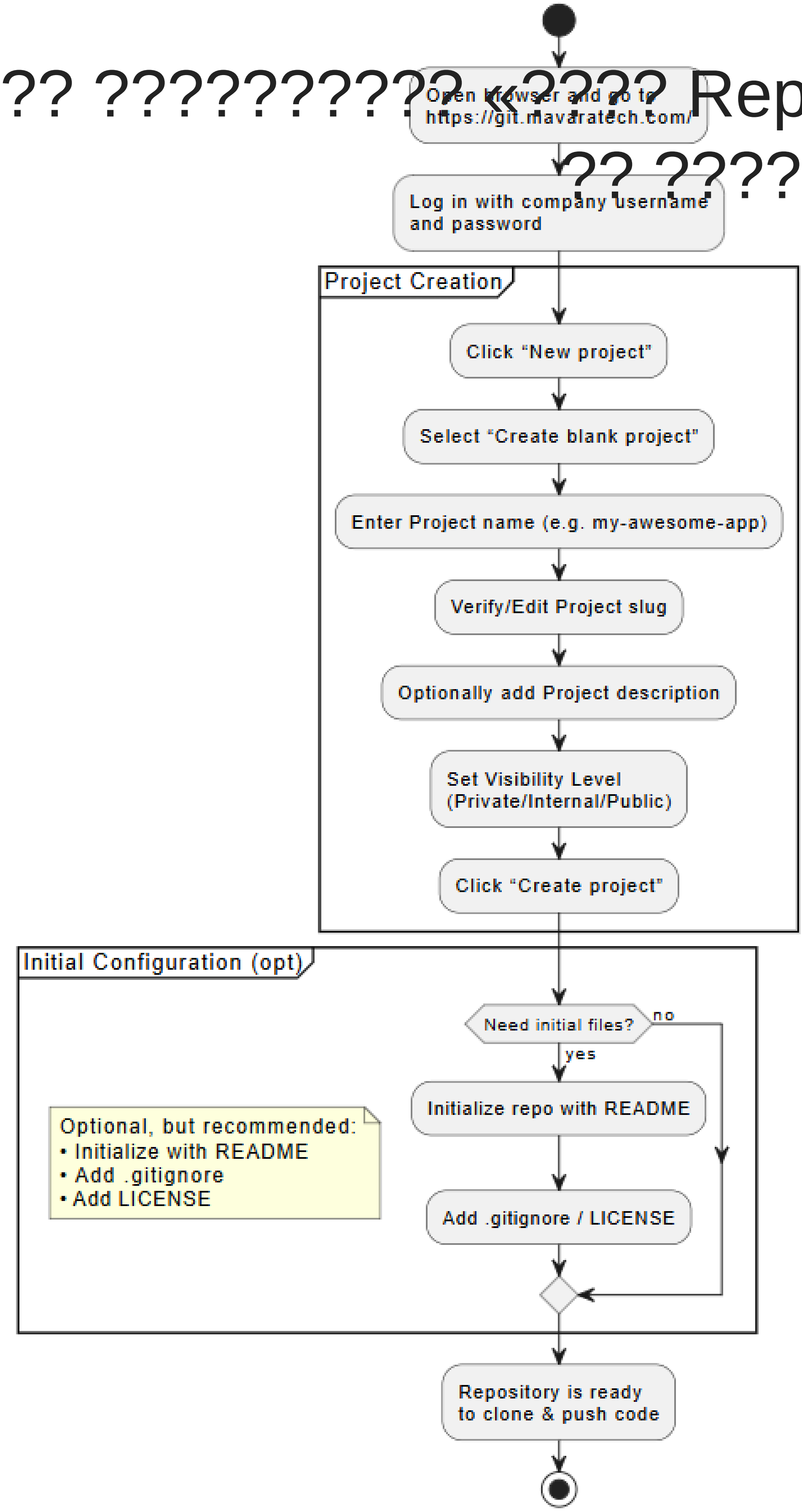
فهرست مراحل


Start
Create repository in company GitLab
Set project config based on project structure
Create & design database
Implement routes, APIs & controllers
Notify PMO that project is done
End

1. **ایجاد Repository در گیتلب شرکت**
   - ورود به `https://git.mavaratech.com` و لاگین با حساب کاربری شرکت
   - کلیک روی **New Project** و انتخاب **Create blank project**
   - وارد کردن نام، توضیحات و تنظیم سطح دسترسی
2. **تنظیم کانفیگ پروژه بر اساس ساختار تعریفشده**
   - ایجاد پوشه `config/` و فایلهای `default.json` و `production.json`
   - ست کردن پارامترهای مثل `database.url`، `baseUrl` و تنظیمات SSO
   - بارگذاری متغیرهای محیطی (Environment Variables)
3. **طراحی و پیادهسازی دیتابیس**
   - مدلسازی ER-Diagram و تعریف جداول، ستونها، PK و FK
   - نوشتن اسکریپتهای DDL و اجرای آنها در محیط توسعه
   - اضافه کردن seed data برای جداول پایه مثل کاربران و نقشها
4. **ایجاد Routeها، API و کنترلرها**
   - تعریف نقشه مسیرها (routes) در `src/routes`
   - پیادهسازی کنترلرها (Controllers) برای هر endpoint در `src/controllers`
   - مستندسازی APIها با OpenAPI/Swagger
5. **اطلاعرسانی به PMO درباره اتمام پروژه**
   - ارسال ایمیل یا تیکت در سیستم PMO مبنی بر پایان کار
   - ارائه گزارش نهایی و لینک به مخزن کد و مستندات تکمیلی

ساخت Repository

????? ??????????? «???? Repository»
?? ?????? ????

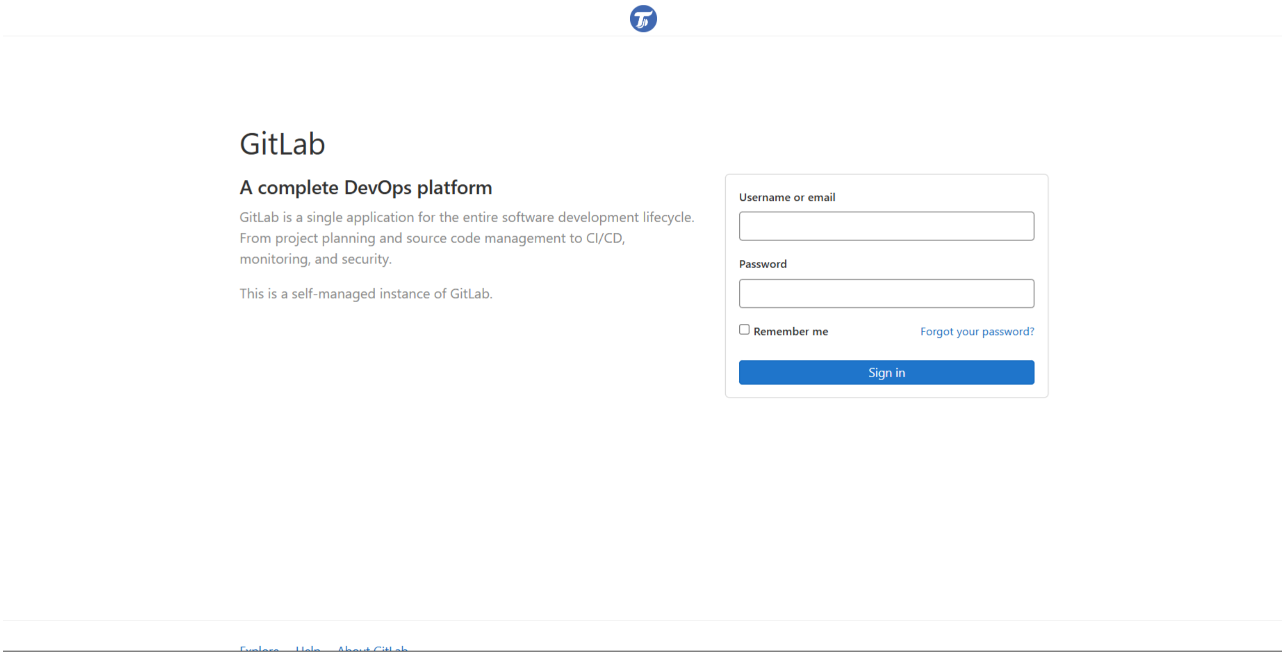

Open browser and go to https://git.mavaratech.com/
Log in with company username and password
Project Creation
Click “New project”
Select “Create blank project”
Enter Project name (e.g. my-awesome-app)
Verify/Edit Project slug
Optionally add Project description
Set Visibility Level (Private/Internal/Public)
Click “Create project”
Initial Configuration (opt)
Need initial files?
no
yes
Initialize repo with README
Add .gitignore / LICENSE
Optional, but recommended:
• Initialize with README
• Add .gitignore
• Add LICENSE
Repository is ready to clone & push code

1. **???? ?? ??????**

????? ?????? ??? ?? ??? ???? ? ?? ???? ??? ?????? ????:

```
https: //git. mavaratech. com/
```

??? ?? ???? ?????? ???? (??? ?????? ? ??? ????) ???? ????.



GitLab

A complete DevOps platform

GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security.

This is a self-managed instance of GitLab.

Username or email

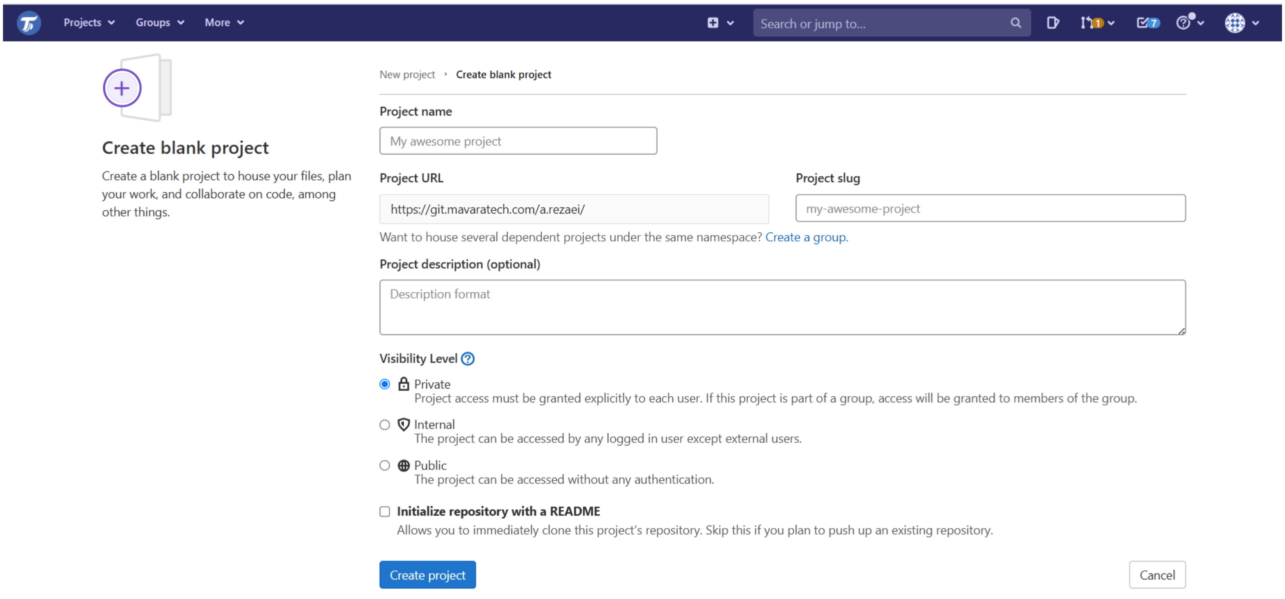
Password

□ Remember me   Forgot your password?

Sign in

Explore Help About GitLab

2. **????? ????? ????**



New project › Create blank project

Create blank project

Create a blank project to house your files, plan your work, and collaborate on code, among other things.

Project name: My awesome project

Project URL: https://git.mavaratech.com/a.rezaei/

Project slug: my-awesome-project

Want to house several dependent projects under the same namespace? Create a group.

Project description (optional): Description format

Visibility Level

● Private
Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

○ Internal
The project can be accessed by any logged in user except external users.

○ Public
The project can be accessed without any authentication.

□ **Initialize repository with a README**
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project   Cancel

- ?? ?? ????? ?? ???? ??? ?? ?? ????? ???? ??? ?????? **New project** ???? ????.
- ??????? **Create blank project** ?? ?????? ????.

3. **???? ???? ??????? ?????**

- **Project name**: ??? ????? ?? ?? ???? ??????? ? ???? ????? (????? `my-awesome-app`) ???? ????.
- **Project slug**: ??????? ?????? ?? ?????? ??? ?? ???? ???? ?????? ????.
- **Project description** (???????): ????? ?????? ???????? ??? ????? ???????.
- **Visibility Level**: ??? ?????? ????? ?? ???? ???? (`Private`? `Internal` ?? `Public`).

4. **???? ??????????**

   ?? ?? ????? ???????? ??? ?????? **Create project** ???? ????. ?????? ?? ??? ?????? ???? ???? ?? ???? ??? ????? ???? ? ???? ?????? ????? ???????.

5. **???????? ????? (???????)**

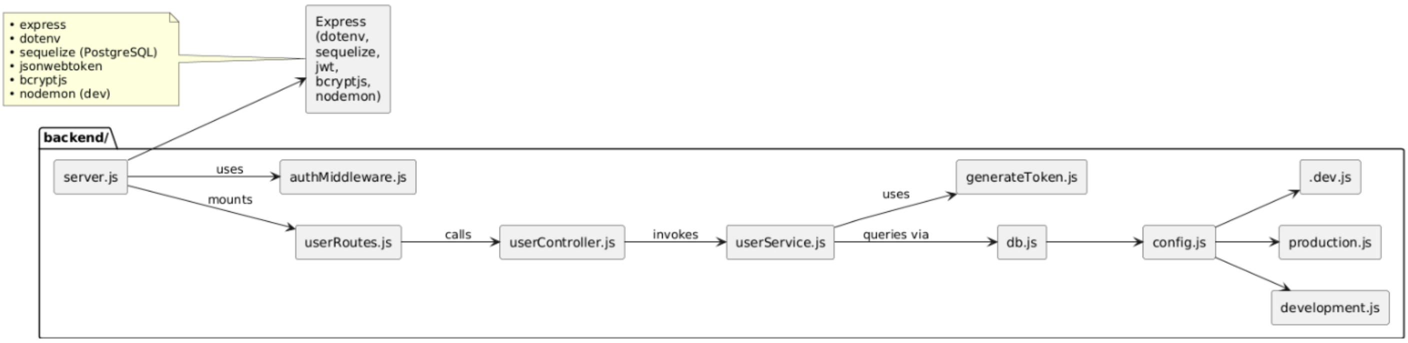
- ?? ???? ????? ?? ???? `README.md`? `.gitignore` ?? ?????? (`LICENSE`) ????? ????.
- ?? ???? ???? **Initialize repository with a README** ?? ???? README ?? ???? ????? ??????.

????? ?????????? ??? ??????? ??? ??? ? ????????? ?? ?? ???? ????? ???? ?? ??? ???? ? ????? ?? ????? ????.

ساختار بندی پروژه

مقدمه

در این راهنما، گام‌به‌گام نحوهٔ شروع یک پروژه فول‌استک با React و Node.js و ساختار پیشنهادی پوشه‌ها و فایل‌ها تشریح شده است. این مستند مناسب تیم‌های توسعه است تا استاندارد یکپارچه‌ای برای ایجاد پروژه‌های جدید داشته باشند.



راه‌اندازی پروژه بکاند (Node.js + Express)

```
mkdir backend
cd backend
npm init -y
npm install express dotenv sequelize jsonwebtoken bcryptjs
npm install --save-dev nodemon
```

ساختار پوشه‌ها

```
backend/
├── config/
│   ├── db.js
│   ├── config.js
│   ├── development.js
│   ├── production.js
│   └── .dev.js
├── controller/
│   └── userController.js
├── routes/
│   └── userRoutes.js
├── service/
│   └── userService.js
├── utils/
│   └── generateToken.js
├── auth/
│   └── authMiddleware.js
├── .env
├── server.js
└── package.json
```

در `server.js` حتماًباید پورت پروژه را با مسئول DevOps تیم هماهنگ کنید.

پیکربندی اتصال به دیتابیس (config)

فایل `db.js`

این کلاس مسئول ایجاد اتصال Sequelize به PostgreSQL است:

```
import { Sequelize } from 'sequelize';
import configs from './config.js';
```

class DBConnection { constructor() { this.db_url = configs.db.url; const options = { dialect: "postgres", timezone: "+03:30", schema: 'schema', }; if (configs.db.useSSL === "true") { options.dialectOptions = { ssl: { require: true, rejectUnauthorized: false } }; } this.db = new Sequelize(this.db_url, options); }

---

getSequelizeInstance() { return this.db; } }

---

const sequelize = new DBConnection().getSequelizeInstance(); export default sequelize;

فایل `config.js`

براساس مقدار `NODE_ENV` یکی از فایل‌های محیطی (development، production، .dev) وارد می‌شود:

```
const env = process.env.NODE_ENV || 'development';
const configs = await import(`./${env}.js`).then(m => m.default);
export default configs;
```

پیکربندی اتصال به دیتابیس (config)

فایل `db.js`

این کلاس مسئول ایجاد اتصال Sequelize به PostgreSQL است:

```


import { Sequelize } from 'sequelize';
import configs from './config.js';
```

class DBConnection { constructor() { this.db_url = configs.db.url; const options = { dialect: "postgres", timezone: "+03:30", schema: 'schema Name', }; if (configs.db.useSSL === "true") { options.dialectOptions = { ssl: { require: true, rejectUnauthorized: false } }; } this.db = new Sequelize(this.db_url, options); }

---

getSequelizeInstance() { return this.db; } }

---

const sequelize = new DBConnection().getSequelizeInstance(); export default sequelize;

فایل `config.js`

براساس مقدار `NODE_ENV` یکی از فایل‌های محیطی (development، production، .dev) وارد می‌شود:

```
const env = process.env.NODE_ENV || 'development';
const configs = await import(`./${env}.js`).then(m => m.default);
export default configs;
```

فایل‌های محیطی

در این فایل‌ها تنظیمات مربوط به هر محیط (production، development، .dev) قرار دارد:

`production.js`

```
export default {
app: {
port: 3000,
apiBaseUrl: process.env.API_BASE_URL || 'url',
ssoBaseUrl: 'url',
paymentBaseUrl: process.env.PAYMENT_BASE_URL || 'url',
useMockUsername: process.env.USE_MOCK_USERNAME || 'false',
mockUsername: process.env.MOCK_USERNAME || '',
},
db: {
url: process.env.DB_URL || 'postgresql://postgreess',
useSSL: process.env.DB_USE_SSL || 'false',
},
redis: {
host: process.env.REDIS_HOST || 'host url',
port: process.env.REDIS_PORT || port,
sentinelNodes: process.env.REDIS_SENTINEL_NODES || '',
sentinelName: process.env.REDIS_SENTINEL_MASTER || 'mymaster',
password: process.env.REDIS_PASSWORD || 'password',
dbNo: process.env.REDIS_DB_NO || 0,
prefix: process.env.REDIS_PREFIX || 'integropia:'
}
};
```

development.js

```
export default {
app: {
port: 3001,
apiBaseUrl: 'BaseUrl that you recive',
useMockUsername: 'true',
mockUsername: process.env.MOCK_USERNAME || '',
},
db: {
url: process.env.DB_URL || 'postgresql:postgressName',
useSSL: 'false',
},
redis: {
host: process.env.REDIS_HOST || '87.248.137.22',
port: process.env.REDIS_PORT || 6379,
sentinelNodes: process.env.REDIS_SENTINEL_NODES || '',
sentinelName: process.env.REDIS_SENTINEL_MASTER || 'mymaster',
password: process.env.REDIS_PASSWORD || 'password',
dbNo: process.env.REDIS_DB_NO || 0,
prefix: process.env.REDIS_PREFIX || 'integropia:',
},
};
```

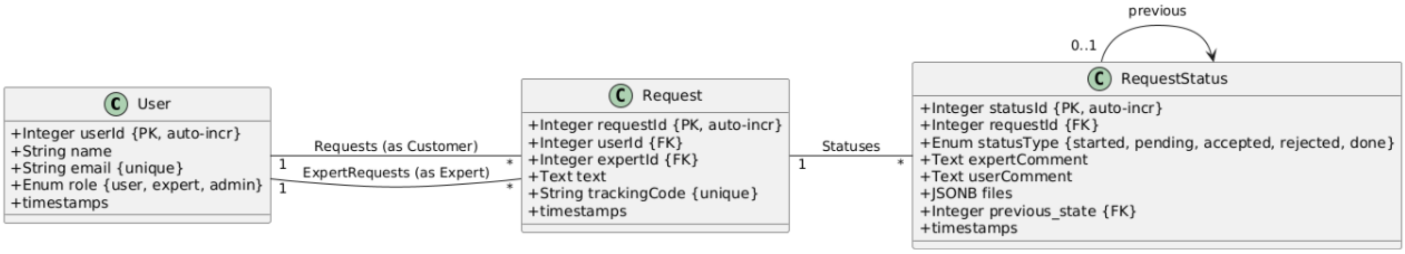
.dev.js

```
export default {
app: {
port: 3000,
apiBaseUrl: process.env.API_BASE_URL || 'url',
ssoBaseUrl: 'url,
paymentBaseUrl: process.env.PAYMENT_BASE_URL || 'url',
useMockUsername: process.env.USE_MOCK_USERNAME || 'false',
mockUsername: process.env.MOCK_USERNAME || '',
},
db: {
url: process.env.DB_URL || 'postgresql:postgress',
useSSL: process.env.DB_USE_SSL || 'false',
},
redis: {
host: process.env.REDIS_HOST || 'host',
port: process.env.REDIS_PORT || port,
sentinelNodes: process.env.REDIS_SENTINEL_NODES || '',
sentinelName: process.env.REDIS_SENTINEL_MASTER || 'mymaster',
password: process.env.REDIS_PASSWORD || 'password',
dbNo: process.env.REDIS_DB_NO || 0,
prefix: process.env.REDIS_PREFIX || 'integropia:'
}
};
```

نکته بسیار مهم: در این بخش باید `secret_key` مینیاپ فرانت خود را در فایلهای `production.js` یا `development.js` قرار دهید و نام مینیاپ در تمام پیکربندیها یکسان باشد.

طراحی ساختار دیتابیس (ER-Diagram)

?? ????? ?? ???? ?? ??? ????? ??????? ????? ??? ???


previous
0..1
User
+Integer userId {PK, auto-incr}
+String name
+String email {unique}
+Enum role {user, expert, admin}
+timestamps
Requests (as Customer)
ExpertRequests (as Expert)
1
1
*
*
Request
+Integer requestId {PK, auto-incr}
+Integer userId {FK}
+Integer expertId {FK}
+Text text
+String trackingCode {unique}
+timestamps
Statuses
1
*
RequestStatus
+Integer statusId {PK, auto-incr}
+Integer requestId {FK}
+Enum statusType {started, pending, accepted, rejected, done}
+Text expertComment
+Text userComment
+JSONB files
+Integer previous_state {FK}
+timestamps

1. **????? ?????? ??????? (ER-Diagram)**
   - ???? ???? ?????????? (Entities): `User`? `Request` ? `RequestStatus`
   - ????? ?????: ?? `User` ???????? ??? `Request` ????? ????? ?? `Request` ??? `RequestStatus`
   - ????? ??????? ???? (PK) ? ????? (FK) ?? ?? ????
2. **????? ?????? Sequelize ???? PostgreSQL**
   - ??? ???????:

```
npm install sequelize pg pg-hstore
```

   - ????? ???? ?????? `src/configs/database.js`:

```
import { Sequelize } from 'sequelize';
const sequelize = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  process.env.DB_PASS,
  {
    host: process.env.DB_HOST,
    dialect: 'postgres',
    logging: false,
  }
);
export default sequelize;
```

3. **????? ??? `User`**
- ????? ??????? ???? ??? `userId`? `name`? `email` ? `role`

```
import { DataTypes } from 'sequelize';
import sequelize from '../configs/database.js';
const User = sequelize.define('User', {
userId: {
type: DataTypes.INTEGER,
autoIncrement: true,
primaryKey: true,
},
name: {
type: DataTypes.STRING(100),
allowNull: false,
},
email: {
type: DataTypes.STRING(150),
allowNull: false,
unique: true,
},
role: {
type: DataTypes.ENUM('user', 'expert', 'admin'),
defaultValue: 'user',
},
}, {
tableName: 'users',
timestamps: true,
});

export default User;
```

4. **????? ??? `Request`**
- ??????? `requestId`? `userId`? `expertId`? `text` ? `trackingCode`

```
import { DataTypes } from 'sequelize';
import sequelize from '../configs/database.js';
import User from './user.js';
import { v4 as uuidv4 } from 'uuid';
const Request = sequelize.define('Request', {
requestId: {
type: DataTypes.INTEGER,
autoIncrement: true,
primaryKey: true,
},
userId: {
type: DataTypes.INTEGER,
references: { model: User, key: 'userId' },
allowNull: true,
},
expertId: {
type: DataTypes.INTEGER,
references: { model: User, key: 'userId' },
allowNull: true,
},
text: {
type: DataTypes.TEXT,
allowNull: false,
},
trackingCode: {
type: DataTypes.STRING(20),
allowNull: false,
unique: true,
defaultValue: () => uuidv4().slice(0, 20),
},
}, {
tableName: 'requests',
timestamps: true,
});

export default Request;
```

5. **????? ??? `RequestStatus`**
- ENUM ????????? ????? ?? `requestId` ? ??????? `previous_state`

```
import { DataTypes } from 'sequelize';
import sequelize from '../configs/database.js';
import Request from './request.js';
```

```
const RequestStatus = sequelize.define('RequestStatus', {
statusId: {
type: DataTypes.INTEGER,
autoIncrement: true,
primaryKey: true,
},
requestId: {
type: DataTypes.INTEGER,
references: { model: Request, key: 'requestId' },
allowNull: false,
},
statusType: {
type: DataTypes.ENUM('started','pending','accepted','rejected','done'),
defaultValue: 'started',
},
expertComment: {
type: DataTypes.TEXT,
defaultValue: '',
},
userComment: {
type: DataTypes.TEXT,
defaultValue: '',
},
files: {
type: DataTypes.JSONB,
allowNull: true,
},
previous_state: {
type: DataTypes.INTEGER,
references: { model: 'request_statuses', key: 'statusId' },
allowNull: true,
},
}, {
tableName: 'request_statuses',
timestamps: true,
hooks: {
beforeCreate: async (status) => {
const last = await RequestStatus.findOne({
where: { requestId: status.requestId },
order: [['statusId','DESC']],
});
if (last) status.previous_state = last.statusId;
}
}
});
export default RequestStatus;
```

6. **????? ???????? (Associations)**

- ?? ???? `src/models/index.js` ?? ?????? ?? ???:

```
import User from './user.js';
import Request from './request.js';
import RequestStatus from './requestStatus.js';
// User ↔ Request
User.hasMany(Request, { foreignKey: 'userId', as: 'Requests' });
Request.belongsTo(User, { foreignKey: 'userId', as: 'Customer' });
User.hasMany(Request, { foreignKey: 'expertId', as: 'ExpertRequests' });
Request.belongsTo(User, { foreignKey: 'expertId', as: 'Expert' });
// Request ↔ RequestStatus
Request.hasMany(RequestStatus, { foreignKey: 'requestId', as: 'Statuses' });
RequestStatus.belongsTo(Request, { foreignKey: 'requestId', as: 'Request' });
export { User, Request, RequestStatus };
```

7. **?????????? ????? ????? (State Pattern)**

- ???? ???? `src/models/state/` ? ????? ???????? ?????:

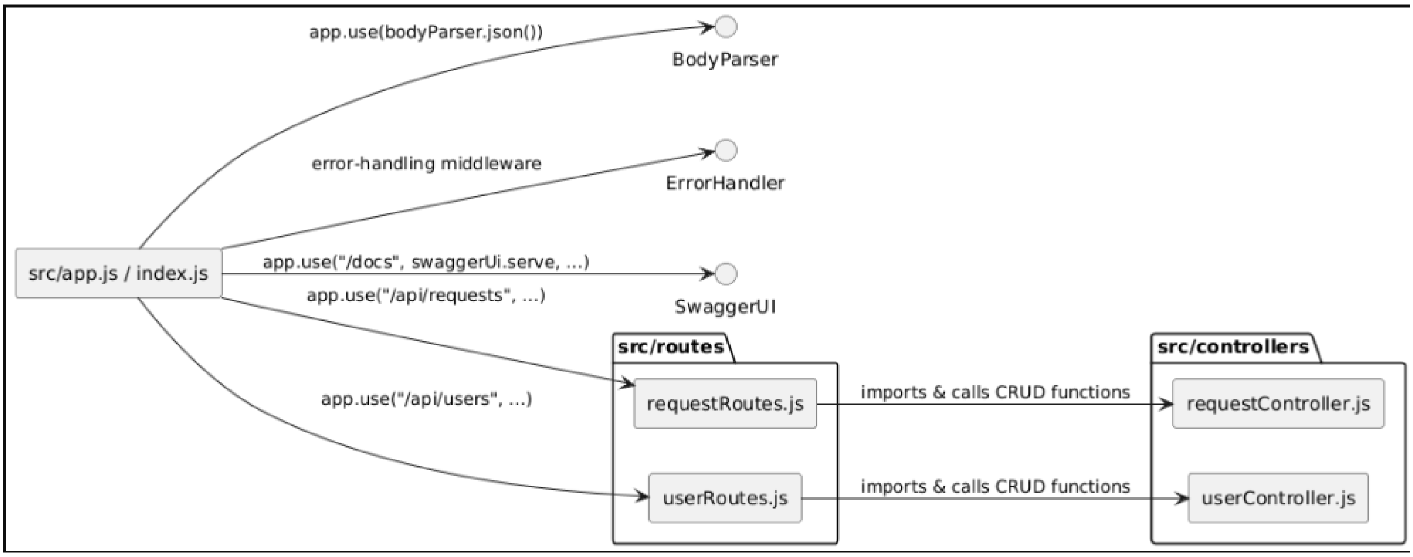
```
import { handleReqRejected } from '../../utils/status.utils.js';
class AcceptedState {
constructor(context) { this.context = context; }
async transitionTo(statusType) {
if (!statusType) throw new Error('Invalid statusType');
if (statusType === 'rejected') {
return await handleReqRejected(this.context);
}
throw new Error(`Cannot transition from accepted to ${statusType}`);
}
}
export default AcceptedState;
```

- ?? ????? Request ???????? ?????:

```
import AcceptedState from './state/acceptedState.js';
import PendingState from './state/pendingState.js';
// …
class RequestService {
constructor(request) {
this.request = request;
this.state = this._getStateInstance(request.currentStatus);
}
_getStateInstance(type) {
switch(type) {
case 'accepted': return new AcceptedState(this.request);
// …
}
}
async changeStatus(toType) {
return this.state.transitionTo(toType);
}
}
```

نوشتن APi, Route , Controller

?? ????? ?? ???? ?? ????? route , controller ????? ??? ???


app.use(bodyParser.json())
BodyParser
error-handling middleware
ErrorHandler
src/app.js / index.js
app.use("/docs", swaggerUi.serve, ...)
SwaggerUI
app.use("/api/requests", ...)
app.use("/api/users", ...)
src/routes
requestRoutes.js
userRoutes.js
imports & calls CRUD functions
imports & calls CRUD functions
src/controllers
requestController.js
userController.js

1. **????? ?????? ??????? ???? Route ? Controller**
   - ?? ???? ????? ???????? `src/routes/` ? `src/controllers/` ?? ??????
   - ?? ??????? (????? `requests` ? `users`) ?? ???? Route ? ?? ???? Controller ????
   - ?????? ?????:

```
src/
├─ controllers/
│   ├─ requestController.js
│   └─ userController.js
└─ routes/
    ├─ requestRoutes.js
    └─ userRoutes.js
```

2. **????? Route?? ?? ??????? ?? Express**
   - ?? `src/routes/requestRoutes.js` ??????? CRUD ?? ????? ????:

```
import { Router } from 'express';
import {
createRequest,
getAllRequests,
getRequestById,
updateRequest,
deleteRequest
} from '../controllers/requestController.js';
const router = Router();

router.post('/', createRequest);
router.get('/', getAllRequests);
router.get('/:id', getRequestById);
router.put('/:id', updateRequest);
router.delete('/:id', deleteRequest);

export default router;
```

3. **????? Controller?? ?? ???? ????**

- ?? `src/controllers/requestController.js` ????? CRUD ?? ????? ????:

```
import Request from '../models/request.js';
export const createRequest = async (req, res, next) => {
try {
const newReq = await Request.create({
userId: req.body.userId,
expertId: req.body.expertId,
text: req.body.text
});
res.status(201).json(newReq);
} catch (err) {
next(err);
}
};

export const getAllRequests = async (_req, res, next) => {
try {
const list = await Request.findAll();
res.json(list);
} catch (err) {
next(err);
}
};

export const getRequestById = async (req, res, next) => {
try {
const item = await Request.findByPk(req.params.id);
if (!item) return res.status(404).json({ message: 'Not found' });
res.json(item);
} catch (err) {
next(err);
}
};

export const updateRequest = async (req, res, next) => {
try {
const [updated] = await Request.update(req.body, {
where: { requestId: req.params.id }
});
if (!updated) return res.status(404).json({ message: 'Not found' });
res.json({ message: 'Updated' });
} catch (err) {
next(err);
}
};

export const deleteRequest = async (req, res, next) => {
try {
const deleted = await Request.destroy({
where: { requestId: req.params.id }
});
if (!deleted) return res.status(404).json({ message: 'Not found' });
res.status(204).end();
} catch (err) {
next(err);
}
};
```

4. **??? ???? Route?? ?? ???????? ????**

- ?? `src/app.js` ?? `index.js` ??? ??? ?? ????? ????:

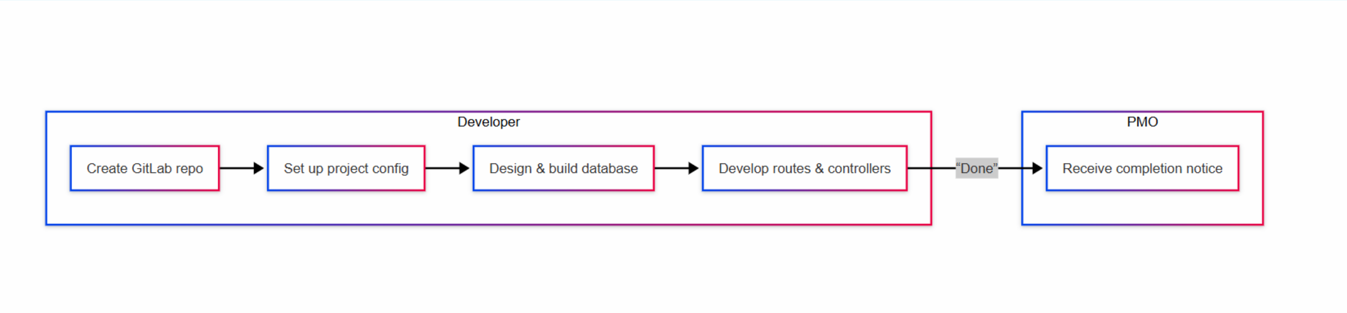
```
import express from 'express';
import bodyParser from 'body-parser';
import requestRoutes from './routes/requestRoutes.js';
import userRoutes    from './routes/userRoutes.js';
const app = express();
app.use(bodyParser.json());

app.use('/api/requests', requestRoutes);
app.use('/api/users',    userRoutes);

// هندلینگ خطا
app.use((err, _req, res, _next) => {
console.error(err);
res.status(500).json({ error: err.message });
});

export default app;
```

نکات مهم پایانی



???? ???

- ????? ?? ??? ????????? ?? ???? ????? ?? ????? ?? ????????? ???? ???? ??? ????.
- ?? ????? DevOps ?????? ???? ?? ??????? ? ??????? ???????? ????? ???.
- ????????? ?? ?? ??? PMO ?? ???? ? ????? ?????? ????????? ??? ?? ????? ????.
- ????????? ?? ?????? ????? ?? ??? ???? ??? QA ???????? ???? ????? ????.
- ??????? ????? ?? ???? ? ???? ???? ???? ?? ?????? ??????? ? ????? ????? ???? ???.