پروژهTodo App

ساینا سرو ر

۱۴۰۱۱۱۵۰۸۵۴۶۱

1.معرفی کلی پروژه

این پروژه یک برنامه **مدیریت وظایف (Todo App)** است که به‌صورت زیر طراحی و پیاده‌سازی شده و شامل:

- **Backend**: API برای مدیریت وظایف
- **Frontend**: رابط کاربری برای نمایش و تعامل
- **Docker**: اجرای سرویس‌ها در کانتینرهای جداگانه
- **Git**: کنترل نسخه حرفه‌ای بر اساس pxull request و feature branch

---

۲ .اهداف پروژه

هدف اصلی توسعه این برنامه شامل موارد زیر است:

- طراحی و پیاده‌سازی API برای مدیریت چرخه حیات وظایف
- ایجاد رابط کاربری ساده و قابل استفاده
- استفاده از **Git** برای کنترل نسخه و همکاری تیمی
- **کانتینری‌سازی** کامل پروژه با Docker
- آماده‌سازی پروژه برای توسعه‌های بعدی

---

۳ .الزامات فنی اجباری انجام‌شده

۳.۱📌 کنترل نسخه باGit

تمام توسعه‌ها در یک **ریپازیتوری مشترک** انجام شده است و ساختار مدیریت نسخه شامل:

- **شاخه اصلی (main)** برای نسخه پایدار
- شاخه‌های مجزا برای توسعه ویژگی‌ها (Feature Branches)
- ادغام تغییرات با استفاده از **Pull Request**
- تاریخچه کامیت‌های واضح، مرحله‌ای و قابل دنبال کردن

این ساختار باعث افزایش کیفیت، خوانایی و قابل پیگیری بودن توسعه شده است.

📌۳.۲ استقرار باDocker

برای اجرای مستقل Backend و دیتابیس، پروژه به‌صورت Dockerized طراحی شده است:

- هر سرویس (Backend و MongoDB) در یک **کانتینر مجزا** اجرا می‌شود
- فایل‌های پیکربندی Docker (Dockerfile) و **Docker Compose** تهیه شده است
- اجرای سیستم با یک دستور ساده:

```
docker compose up --build
```

- که Backend و MongoDB را همزمان اجرا می‌کند

این روش باعث می‌شود پروژه در هر محیطی بدون وابستگی خاص قابل اجرا باشد.

---

۴ .معماری و ساختار پروژه

📁ساختار کلی

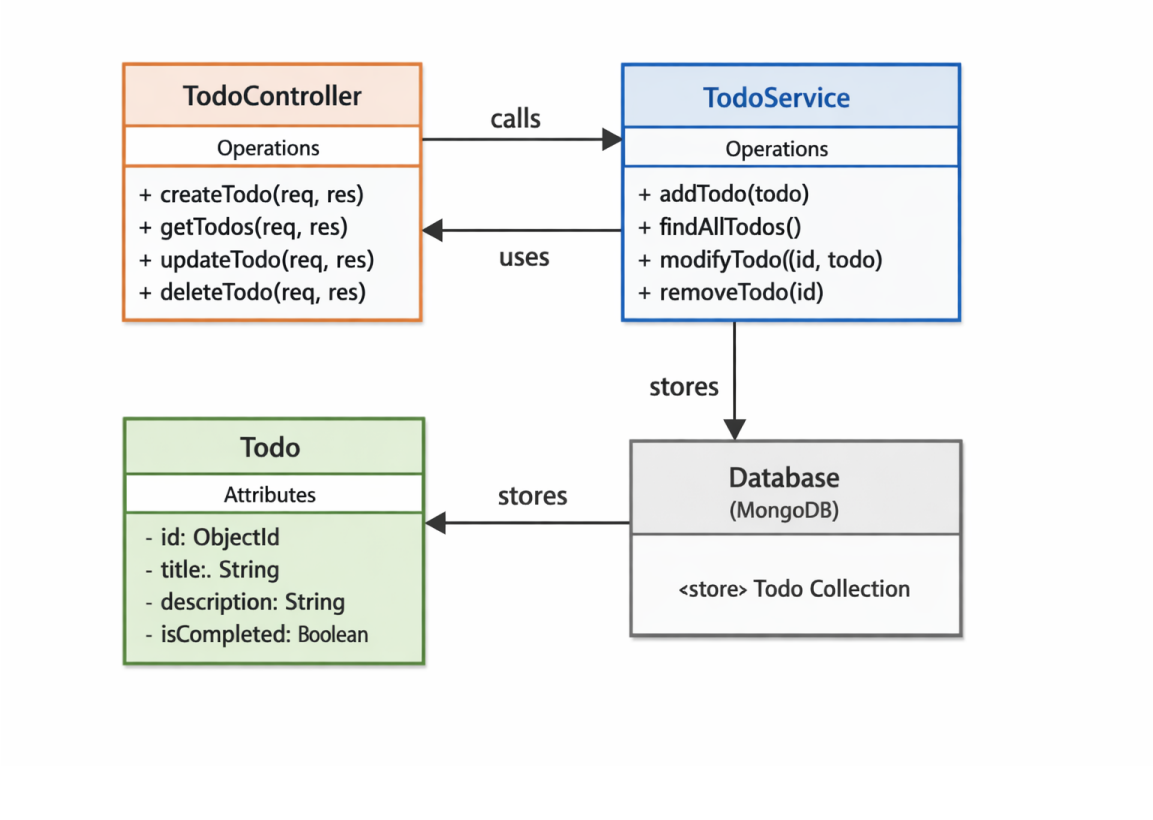
```
todo-app/
├── backend/
│   ├── src/
│   │   ├── models/
│   │   ├── routes/
│   │   ├── config/
│   │   └── index.js
│   ├── Dockerfile
│   └── package.json
│
├── frontend/
│   ├── index.html
│   ├── styles.css
```

```
    └── script.js

├── docker-compose.yml
└── README.md
```

ساختار پروژه به‌صورت جداگانه برای Backend و Frontend طراحی شده و هر بخش مسئولیت مشخصی دارد.

Uml of project:



---

۵ .فناوری‌های استفاده‌شده

🔷 Backend

- Node.js
- Express.js (برای API)
- MongoDB (دیتابیس NoSQL)
- Mongoose (برای مدل‌سازی داده‌ها)

🔷 Frontend

- HTML
- CSS
- JavaScript

🔷 DevOps

- Docker
- Docker Compose
- Git & GitHub

این انتخاب‌ها باعث ساخت یک معماری منعطف و قابل گسترش شده‌اند.

---

۶ – api و قابلیت های backend

Backendپروژه شاملAPI های پایه‌ای برای مدیریتTodo هاست:

وظیفه	مسیر API	روش HTTP
دریافت همه وظایف	/api/todos	**GET**
ایجاد یک وظیفه جدید	/api/todos	**POST**
به‌روزرسانی یک وظیفه	/api/todos/:id	**PUT**
حذف یک وظیفه	/api/todos/:id	**DELETE**

این APIها مطابق استانداردهای REST طراحی شده‌اند و امکان توسعه راحت‌تر را فراهم می‌کنند.

---

۷ .روند انجام پروژه

روند توسعه پروژه به‌صورت مرحله‌ای و حرفه‌ای انجام شده است:

1. **بررسی الزامات تکلیف و طراحی کلی**
2. **طراحی معماری پروژه و تفکیک ساختار frontend/backend**
3. **پیاده‌سازی API با Express.js**
4. **طراحی UI ساده با HTML/CSS/JS**
5. **کانتینری‌سازی سرویس‌ها با Docker**
6. **مدیریت نسخه با Git و شاخه‌سازی مناسب**
7. **اتمام و مستندسازی نهایی**

---

۸ .نتیجه‌گیری

این پروژه نمونه‌ای استاندارد از پیاده‌سازی یک برنامه Full-Stack با توجه به الزامات درسی است که:

✔ Dockerرا برای استقرار سرویس‌ها به‌کار گرفته

✔ APIو frontend را به‌صورت منسجم و قابل استفاده طراحی کرده است