TechCareer Fullstack Development Bootcamp

Todolist Capstone Project

Spring Boot + React

İSMAİL HİLMİ ÖZÇELİK

Todolist Proje Genel Layout

- **Görev ekle**
  - Görev ismi girmek için bir input
  - Görevi eklemek için buton
- **Filtreleme Butonları**
  - **“**All”: Tüm görevleri listeler
  - "Completed": Tamamlanan görevleri listeler
  - "Todo": Yapılacak görevleri listeler

- **Görevlerin listesi**
  - Görevler eklendikten sonra listelenmeli.
  - **Her görevin sahip olması gereken özellikler:**
    - “checkbox: İlgili görevi tamamlandı olarak işaretler
    - “edit" : İlgili görevi düzenler
    - "remove" : İlgili görevi siler

- **Toplu Sil Butonları**
  - "Delete Done Tasks": Tamamlanmış olarak işaretlenen tüm görevleri siler
  - "Delete All Tasks": Tamamlanıp tamamlandığı fark etmeksizin tüm görevleri siler.

Todolist Proje Altyapısının oluşturulması

Spring Initializr kullanarak projenin altyapısını oluşturduk. Altyapıyı oluştururken Maven temelini kullandık. Projeyi oluştururken ihtiyaç duyduğumuz dependencyleri ekledik.

Proje altyapısını oluşturduktan sonra IntelliJ Idea üzerinden projemizi açtık.

Dependencyleri kontrol ettikten sonra application. properties kısmına gerek duyduğumuz kodları ekledik. (H2DB Konfigürasyonu)

Proje Mimarisi
kullanılan bazı dependencyler

Maven (Proje Mimarisi)
Spring Dev Tools
Spring Data JPA
H2DB

Toollar
IDEler

IntelliJ Idea
VS Code

Backend - Class/Interface

TODOLIST

Backendimizde Model "Entity" yapısıyla görevlerimizin yapısını oluşturup veri tutarlılığını sağladık. "Repository" katmanı ile belirli depolama ve alma işlemlerini gerçekleştirerek doğrudan veritabanıyla arayüz oluşturduk. "Service" katmanında temel iş mantığını belirleyerek verilerin nasıl işleneceğine ve dönüştürüleceğine karar verdik. "Controller" ile gelen istekleri, "Service" katmanıyla olan arayüzü, uygun responseları geri göndermeyi vb. harici etkileşimleri yönettik.

01 **Model - Entity**
Veritabanıyla eşleşen değişkenleri/verileri içerir

02 **Repository**
JPA aracılığıyla varsayılan CRUD işlemlerini sağlayan arayüzdür

03 **Service**
İş mantığını ve repository ile olan arayüzü içerir

04 **Controller**
Frontend'i Backend'e bağlayarak APi isteklerini yönetir

05 **Config**
CORS (kökenler arası kaynak paylaşımını) sağlar. **http://localhost:8080** <--> **http://localhost:3000**

Backend - Class/Interface

TODOLIST

Entity

```java
package com.todoreact.project.model;

import ...

19 usages  ismail h. ozcelik
@Entity
public class Task {

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;
3 usages
private String name;
3 usages
private boolean completed;

ismail h. ozcelik
public Task(){
}

no usages  ismail h. ozcelik
public Task(Long id, String name, boolean completed) {
```

Service

```java
public class TaskService {

@Autowired
private TaskRepository taskRepository;

1 usage  ismail h. ozcelik
public List<Task>getAllTasks() { return taskRepository.findAll(); }

2 usages  ismail h. ozcelik
public Optional<Task>getTaskById(Long id){
return taskRepository.findById(id);
}

2 usages  ismail h. ozcelik
public Task saveTask(Task task){
return taskRepository.save(task);
```

Repository

```java
package com.todoreact.project.repository;

import ...

2 usages  ismail h. ozcelik
public interface TaskRepository extends JpaRepository<Task, Long> {

    2 usages  ismail h. ozcelik
    List<Task>findByCompleted(boolean status);
}
```

Backend - Class/Interface

TODOLIST

Controller

```java
package com.todoreact.project.controller;

import ...
 ismail h. ozcelik
@RestController
@RequestMapping("/api/tasks")
public class TaskController {

    @Autowired
    private TaskService taskService;

     ismail h. ozcelik
    @GetMapping
    public List<Task> getAllTasks() {
        return taskService.getAllTasks();
    }

     ismail h. ozcelik
    @GetMapping("/{id}")
    public Task getTaskById(@PathVariable Long id) {
        return taskService.getTaskById(id).orElse( other: null);
    }
```

Config

```java
package com.todoreact.project;


import ...

 ismail h. ozcelik
@Configuration
public class WebConfig implements WebMvcConfigurer {

    no usages  ismail h. ozcelik
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping( pathPattern: "/**")
                .allowedMethods("*")
                .allowedOrigins("http://localhost:3000");
    }
}
```

Frontend - React

T O D O L I S T

Backendimizin sorunsuz çalıştığından emin olduktan sonra frontend aşamasına geçtik. Gerekli kurulumları yaptıktan sonra (Node.js) proje dizininde **npx create-react-app** komutuyla React projesi oluşturduk.

- Bileşenleri ve diğer yardımcı programları içeren bir dizin yapısı oluşturarak frontendimizi düzenledik.
- App, TaskList, TaskInput, Delete Buttons, Filter Buttons vb. bileşenleri oluşturduk.

- Görevleri ve operasyonları (Ekleme, Düzenleme, Silme vb.) yönetmek için React'ın useState ve useEffect özelliklerini kullandık.

- Axios kullanarak frontend'in backend'den veri göndermesine ve almasına izin verdik.
- Tasklere yönelik CRUD işlemlerini uygun endpoint noktalarına HTTP istekleri yaparak uyguladık.

- Kodu temiz ve bakımı kolay tutmak için daha küçük, daha yönetilebilir parçalara ayırdık. Örneğin silme butonlarını kendi bileşenlerine ayırdık.

Frontend – React

TODOLIST

App.js

```
import ...

5+ usages  ismail h. ozcelik
function App() {
    const [tasks : any[] , setTasks] = useState( initialState: []);
    const [filter : string , setFilter] = useState( initialState: "all");

    useEffect( effect: () : void  => {
        axios.get( url: 'http://localhost:8080/api/tasks')
            .then(response : AxiosResponse<any>  => {
                setTasks(response.data);
            });
    }, deps: []);

    1 usage  ismail h. ozcelik
    const handleAddTask = (name) : void  => {
        axios.post( url: 'http://localhost:8080/api/tasks', data: { name, complet
            .then(response : AxiosResponse<any>  => {
                setTasks( value: [...tasks, response.data]);
            });
    };

    1 usage  ismail h. ozcelik
```

TaskList.js

```
import React, { useState } from 'react';
import './TaskList.css';

3 usages  ismail h. ozcelik
function TaskList({ tasks, onToggle, onDelete, onEdit }) {
    const [editTaskId, setEditTaskId] = useState( initialState: null)
    const [editedName : string , setEditedName] = useState( initialStat

    return (
        <div className="tasks-list"...>

    );
}

2 usages  ismail h. ozcelik
export default TaskList;
```

Projeyi ayağa kaldırma

TODOLIST

Önce backend kısmımızı çalıştırıyoruz. **localhost:8080** alanında projeyi sorunsuzca ayağa kaldırdıktan sonra;
Terminal üzerinden: **cd frontend,** ardından
npm start komutuyla projemizi açıyoruz.
localhost:3000

Bu aşamadan sonra bir sonraki sayfadaki ekranı görmeli ve işlemlerimizi yapmaya başlayabilmeliyiz.

Proje github linki: https://github.com/ishilmiozcelik/techcareer_toDoCapstone

Todolist Önyüz

TODOLIST