



TechCareer | Full Stack Developer
Bootcamp Final Projesi

Full Stack ToDo List

Uygulaması

Spring Boot + React

İremnaz Yolcu | 02.09.2023



System.out.println("Begin");

Todo List Uygulaması

Amaç

Kullanıcıların günlük görevlerini ve yapılacaklar listelerini düzenlemelerine yardımcı olan bir web tabanlı ToDoList uygulaması geliştirmek.

Hedef

- Kullanıcıların günlük yaşamlarını daha iyi organize etmelerine yardımcı olmak,
- görevleri daha etkili bir şekilde yönetmelerini sağlamak,
- daha verimli olmalarına katkıda bulunmak.

React - Front End Proje Yapısı

The screenshot shows a file explorer interface with the following project structure:

- FRONTEND**
 - node_modules**
 - public**
 - favicon.ico
 - index.html
 - logo192.png
 - logo512.png
 - manifest.json
 - robots.txt
 - src**
 - component**
 - todos**
 - ToDoCreate.jsx
 - ToDoList.jsx
 - ToDoUpdate.jsx
 - ToDoView.jsx
 - footer.css
 - Footer.jsx
 - Header.jsx
 - Main.jsx

- img**
 - moon.jpg
- services**
 - ToDoApi.js
 - App.css
 - App.js
 - App.test.js
 - index.css
 - index.js
 - logo.svg
 - reportWebVitals.js
 - setupTests.js
 - TodoRouter.jsx
 - .gitignore
 - package-lock.json
 - package.json
 - README.md

OUTLINE
TIMELINE

Spring Boot - Back End Proje Yapısı

```
src
└── main
    ├── java
    │   └── com.iremnazyolcu
    │       ├── assist
    │       │   ├── FrontEnd
    │       │   └── PostmanPersist
    │       ├── bean
    │       │   ├── ModelMapperBean
    │       │   ├── OpenApiConfigBean
    │       │   └── PasswordEncoderBean
    │       ├── business
    │       │   ├── dto
    │       │       └── ToDoDto
    │       │   ├── services
    │       │       └── impl
    │       │           └── ToDoServiceImpl
    │       └── ToDoService
    └── controller.api
        └── ToDoController
    └── data
        └── entity
            └── ToDoEntity
    └── repository
        └── ToDoRepository
    └── error
        ├── ApiResult
        └── CustomErrorHandlerWebRequest
    └── exception
        ├── IremnazYolcuException
        ├── TodoBadRequestException
        ├── TodoCreatedException
        ├── TodoNotFoundException
        ├── TodoUnAuthorizedException
        └── TechCareerFullStackToDoListApplication
```

Back End - ToDoEntity.java

```
© ToDoEntity.java ×  
1 package com.iremnazyolcu.data.entity;  
2  
3 > import ...  
4  
5 // LOMBOK  
6 21 usages  
7 @Data  
8  
9 // ENTITY  
10 @Entity  
11 // Hibernate will create this table in a database with the following entities :  
12 @Table(name = "todos")  
13 public class ToDoEntity implements Serializable {  
14  
15     // Serialization  
16     public static final Long serialVersionUID = 1L;  
17  
18     // ID  
19     @Id  
20     @GeneratedValue(strategy = GenerationType.IDENTITY)  
21     @Column(name = "todo_id", unique = true, nullable = false, insertable = true, updatable = false)  
22     private Long id;  
23  
24     // TITLE  
25     @Column(nullable = false)  
26     private String title;  
27  
28     // DESCRIPTION  
29     @Column(nullable = false)  
30     private String description;  
31  
32     // COMPLETED  
33     private boolean completed;
```

Back End - ToDoRepository.java

```
ToDoRepository.java ×  
1 package com.iremnazyolcu.data.repository;  
2  
3 > import ...  
4  
5 3 usages  
6 @Repository  
7 // ToDoRepository will be able to perform CRUD operations on the Entity specified in place of entityName.  
8 // JpaRepository<entityName, primaryKeyType>  
9 public interface ToDoRepository extends JpaRepository<ToDoEntity, Long> {  
10     2 usages  
11     List<ToDoEntity> findAllByCompleted(boolean completed);  
12 }  
13  
14  
15  
16
```

Back End - ToDoDto.java

ToDoDto.java ×

```
1 package com.iremnazyolcu.business.dto;
2
3 > import ...
10
11 // LOMBOK
12 // 48 usages
12 @Data
13 public class ToDoDto implements Serializable {
14
15     // Serialization
16     public static final Long serialVersionUID = 1L;
17
18     // ToDoDto attributes :
19
20     // ID
21     private Long id;
22
23     // TITLE
24     private String title;
25
26     // DESCRIPTION
27     private String description;
28
29     // COMPLETED
30     private boolean completed;
31
32 }
33
```

Back End - ToDoService.java

```
ToDoService.java ×  
11  // CREATE  
12  ● 1 usage 1 implementation  
13  ToDoDto createTodo(ToDoDto todoDto);  
14  
15  // GET  
16  ● 1 usage 1 implementation  
17  ToDoDto getTodo(Long id);  
18  
19  // GET ALL  
20  ● 2 usages 1 implementation  
21  List<ToDoDto> getAllTodos();  
22  
23  // UPDATE  
24  ● 1 usage 1 implementation  
25  ToDoDto updateTodo(ToDoDto todoDto, Long id);  
26  
27  // DELETE  
28  ● 1 usage 1 implementation  
29  void deleteTodo(Long id);  
30  
31  // COMPLETE  
32  ● 1 usage 1 implementation  
33  ToDoDto completeTodo(Long id);  
34  
35  // INCOMPLETE  
36  ● 1 usage 1 implementation  
37  ToDoDto inCompleteTodo(Long id);  
38  
39  // DELETE ALL  
40  ● 1 usage 1 implementation  
41  String deleteAllTodos();  
42  
43  // GET COMPLETED  
44  ● 1 usage 1 implementation  
45  List<ToDoDto> getCompletedTodo(boolean completed);  
46  
47  // DELETE COMPLETED  
48  ● 1 usage 1 implementation  
49  void deleteCompletedTodo();
```

Back End - ToDoServiceImpl.java

```
ToDoServiceImpl.java
14
15  @Service
16  // Constructor based dependency injection
17  @AllArgsConstructor
18  // ToDoServiceImpl class implements the ToDoService interface and its methods.
19  public class ToDoServiceImpl implements ToDoService {
20
21      // Injection
22      // This ToDoServiceImpl requires ToDoRepository as a dependency.
23      private ToDoRepository todoRepository;
24      private ModelMapper modelMapper;
25
26      // CREATE
27      // usage
28      @Override
29      public ToDoDto createTodo(ToDoDto todoDto) {
30
31          // Model Mapper Library is one of the mapping library that we can use to automatically convert
32          // DTO into Entity and Entity into DTO.
33          ToDoEntity todo = modelMapper.map(todoDto, ToDoEntity.class);
34          ToDoEntity savedTodo = todoRepository.save(todo);
35          ToDoDto savedTodoDto = modelMapper.map(savedTodo, ToDoDto.class);
36          return savedTodoDto;
37      }
38
39      // GET
40      // usage
41      @Override
42      public ToDoDto getTodo(Long id) {
43
44          // retrieve the ToDo entity object from the database :
45          ToDoEntity todo = todoRepository.findById(id)
46              .orElseThrow(() -> new TodoNotFoundException("Todo not found with id:" + id));
47
48          // getTodo() method return type is ToDoDto
49          // so we should convert this ToDo entity object into ToDoDto :
50          return modelMapper.map(todo, ToDoDto.class);
51      }
52  }
```

Back End - ToDoServiceImpl.java

```
52     // GET ALL
53     2 usages
54     @Override
55     public List<ToDoDto> getAllTodos() {
56         List<ToDoEntity> todos = todoRepository.findAll();
57
58         return todos.stream().map(todo -> modelMapper.map(todo, ToDoDto.class))
59             .collect(Collectors.toList());
60     }
61
62     // UPDATE
63     1 usage
64     @Override
65     public ToDoDto updateTodo(ToDoDto todoDto, Long id) {
66
67         ToDoEntity todo = todoRepository.findById(id)
68             .orElseThrow(() -> new TodoNotFoundException("Todo not found with id : " + id));
69         todo.setTitle(todoDto.getTitle());
70         todo.setDescription(todoDto.getDescription());
71         todo.setCompleted(todoDto.isCompleted());
72
73         ToDoEntity updatedTodo = todoRepository.save(todo);
74
75         return modelMapper.map(updatedTodo, ToDoDto.class);
76     }
77
78     // DELETE
79     1 usage
80     @Override
81     public void deleteTodo(Long id) {
82
83         ToDoEntity todo = todoRepository.findById(id)
84             .orElseThrow(() -> new TodoNotFoundException("Todo not found with id : " + id));
85
86         todoRepository.deleteById(id);
87     }
88 }
```

Back End - ToDoServiceImpl.java

```
86     // COMPLETE
87     1 usage
88     @Override
89     public ToDoDto completeTodo(Long id) {
90
91         ToDoEntity todo = todoRepository.findById(id)
92             .orElseThrow(() -> new TodoNotFoundException("Todo not found with id : " + id));
93
94         todo.setCompleted(Boolean.TRUE);
95
96         ToDoEntity updatedTodo = todoRepository.save(todo);
97
98         return modelMapper.map(updatedTodo, ToDoDto.class);
99     }
100
101    // INCOMPLETE
102    1 usage
103    @Override
104    public ToDoDto inCompleteTodo(Long id) {
105
106        ToDoEntity todo = todoRepository.findById(id)
107            .orElseThrow(() -> new TodoNotFoundException("Todo not found with id : " + id));
108
109        todo.setCompleted(Boolean.FALSE);
110
111        ToDoEntity updatedTodo = todoRepository.save(todo);
112
113        return modelMapper.map(updatedTodo, ToDoDto.class);
114    }
115
116    // DELETE ALL
117    1 usage
118    @Override
119    public String deleteAllTodos() {
120        todoRepository.deleteAll();
121        return "Total number of deleted tasks: " + getAllTodos().size();
122    }
```

Back End - ToDoServiceImpl.java

```
121     // GET COMPLETED
122     1usage
123     @Override
124     public List<ToDoDto> getCompletedTodo(boolean completed) {
125         List<ToDoEntity> completedTodos = todoRepository.findAllByCompleted(completed);
126
127         return completedTodos.stream() Stream<ToDoEntity>
128             .map(entity -> modelMapper.map(entity, ToDoDto.class)) Stream<ToDoDto>
129             .collect(Collectors.toList());
130     }
131
132     // DELETE COMPLETED
133     1usage
134     @Override
135     public void deleteCompletedTodo() {
136         List<ToDoEntity> completedTodos = todoRepository.findAllByCompleted(true);
137         for (ToDoEntity completedTodo : completedTodos) {
138             todoRepository.delete(completedTodo);
139         }
140     }
```

Back End - ToDoController.java

© ToDoController.java ×

```
1 package com.iremnazyolcu.controller.api;
2
3 > import ...
4
5 // The @RestController annotation designates this class as a Spring MVC REST controller.
6 // Within this Spring MVC REST controller, we can define the REST APIs.
7 @RestController
8 // @RequestMapping annotation establishes the base URL for all the REST APIs within the ToDoController class.
9 @RequestMapping("api/todos")
10 // Constructor based dependency injection
11 @AllArgsConstructor
12 @CrossOrigin(origins = FrontEnd.REACT_URL)
13 public class ToDoController {
14
15     // Injection
16     // This ToDoController requires ToDoService as a dependency.
17     private ToDoService todoService;
18
19     // Constructor based dependency injection
20
21     // CREATE
22     // http://localhost:4444/api/todos/create
23     @PostMapping("/create")
24     public ResponseEntity<ToDoDto> createTodo(@RequestBody ToDoDto todoDto) {
25         ToDoDto createdTodo = todoService.createTodo(todoDto);
26         return new ResponseEntity<>(createdTodo, HttpStatus.CREATED);
27     }
28
29     // GET
30     // http://localhost:4444/api/todos/list/{id}
31     @GetMapping("/{id}")
32     public ResponseEntity<ToDoDto> getTodo(@PathVariable("id") Long id) {
33         ToDoDto todo = todoService.getTodo(id);
34         if (todo != null) {
35             return new ResponseEntity<>(todo, HttpStatus.OK);
36         } else {
37             return new ResponseEntity<>(HttpStatus.NOT_FOUND);
38         }
39     }
40 }
```

Back End - ToDoController.java

```
49     // GET ALL
50     // http://localhost:4444/api/todos/list/all
51     @GetMapping(value = "/list/all")
52     public ResponseEntity<List<ToDoDto>> getAllTodos() {
53         List<ToDoDto> todos = todoService.getAllTodos();
54         return new ResponseEntity<>(todos, HttpStatus.OK);
55     }
56
57     // UPDATE
58     // http://localhost:4444/api/todos/update/1
59     @PutMapping(value = "/update/{id}")
60     public ResponseEntity<ToDoDto> updateTodo(@RequestBody ToDoDto todoDto, @PathVariable("id") Long id) {
61         ToDoDto updatedTodo = todoService.updateTodo(todoDto, id);
62         if (updatedTodo != null) {
63             return new ResponseEntity<>(updatedTodo, HttpStatus.OK);
64         } else {
65             return new ResponseEntity<>(HttpStatus.NOT_FOUND);
66         }
67     }
68
69     // DELETE
70     // http://localhost:4444/api/todos/delete/1
71     @DeleteMapping(value = "/delete/{id}")
72     public ResponseEntity<String> deleteTodo(@PathVariable("id") Long id) {
73         todoService.deleteTodo(id);
74         return ResponseEntity.ok( body: "Todo deleted successfully!");
75     }
76
77     // DELETE ALL
78     // http://localhost:4444/api/todos/delete/all
79     @DeleteMapping(value = value = "/delete/all")
80     public ResponseEntity<String> allDeleteTodo() { return ResponseEntity.ok(todoService.deleteAllTodos()); }
81
82
83
```

Back End - ToDoController.java

```
84 // COMPLETE
85 // http://localhost:4444/api/todos/complete/1
86 @PatchMapping("/{id}")
87 public ResponseEntity<ToDoDto> completeTodo(@PathVariable("id") Long id) {
88     ToDoDto updatedTodo = todoService.completeTodo(id);
89     return ResponseEntity.ok(updatedTodo);
90 }
91
92 // GET COMPLETED
93 // http://localhost:4444/api/todos/completed/true
94 @GetMapping("/{completed}")
95 public ResponseEntity<List<ToDoDto>> getCompletedTodos(@PathVariable boolean completed) {
96     List<ToDoDto> completedTodos = todoService.getCompletedTodo(completed);
97     return ResponseEntity.ok(completedTodos);
98 }
99
100 // DELETE COMPLETED
101 // http://localhost:4444/api/todos/delete/completed
102 @DeleteMapping("/{completed}")
103 public ResponseEntity<String> deleteCompletedTodo() {
104     todoService.deleteCompletedTodo();
105     return ResponseEntity.ok( body: "All completed todos deleted successfully!");
106 }
107
108 // INCOMPLETE
109 // http://localhost:4444/api/todos/incomplete/1
110 @PatchMapping("/{id}")
111 public ResponseEntity<ToDoDto> inCompleteTodo(@PathVariable("id") Long id) {
112     ToDoDto updatedTodo = todoService.inCompleteTodo(id);
113     return ResponseEntity.ok(updatedTodo);
114 }
115
```

Database Yapısı

todos		
PK	id	Long
	title	String
	description	String
	completed	boolean

English

Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded)

Setting Name: Generic H2 (Embedded)

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:file:/memory.persist/finalproject

User Name: sa

Password:

SQL statement:

SELECT * FROM TODOS |

SELECT * FROM TODOS;

TODO_ID	COMPLETED	DESCRIPTION	TITLE
1	TRUE	demo description techcrr	demo title techcrr
2	FALSE	demo description techcrr 2	demo title techcrr 2

(2 rows, 3 ms)

Edit

Demo Todo List Gösterim

http://localhost:3000/todos/list/all

Todo List

Todo List						
ID	TITLE	DESCRIPTION	COMPLETED	UPDATE	VIEW	DELETE
1	demo title techcrr	demo description techcrr				
2	demo title techcrr 2	demo description techcrr 2				



TechCareer - Full Stack
Developer Bootcamp
Final Projesi



iremnaz Yolcu

-  iremnazyolcu@gmail.com
-  <https://github.com/iremnazyolcuu>
-  <http://www.linkedin.com/in/iremnazyolcu>

System.out.println("The End");