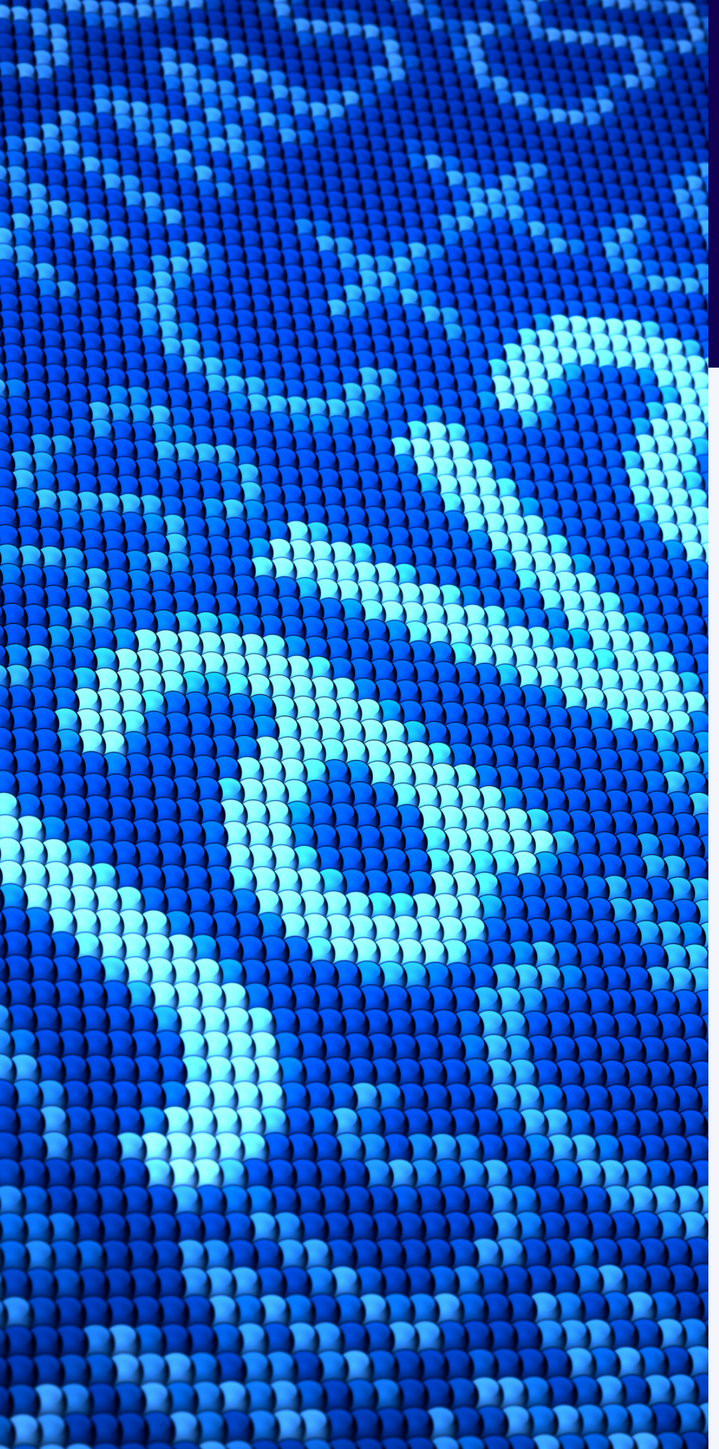
TechCareer | Full Stack Developer
Bootcamp Final Projesi

Full Stack ToDo List Uygulaması Spring Boot + React

İremnaz Yolcu | 02.09.2023

```
System.out.println("Begin");
```



Todo List Uygulaması

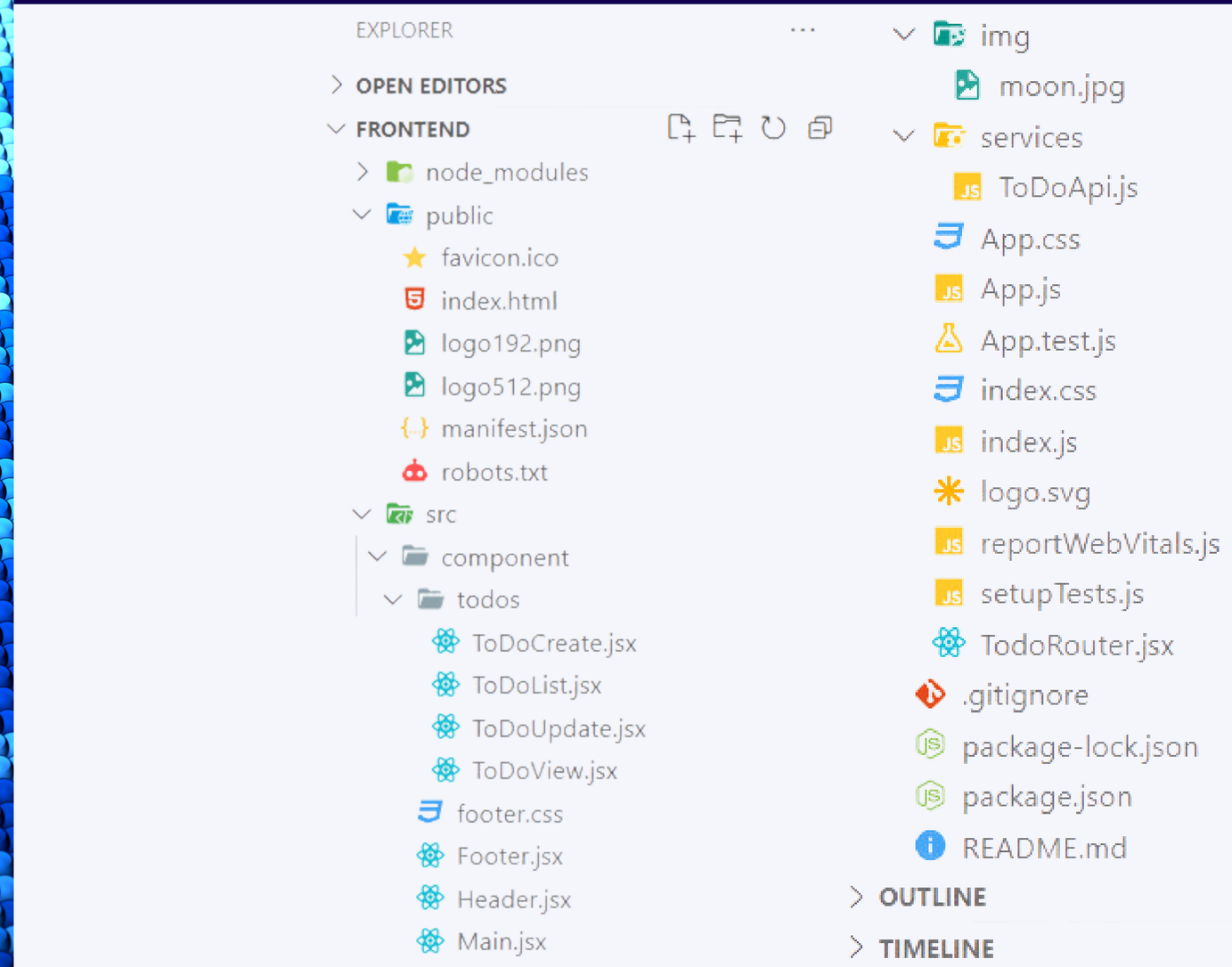
Amaç

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

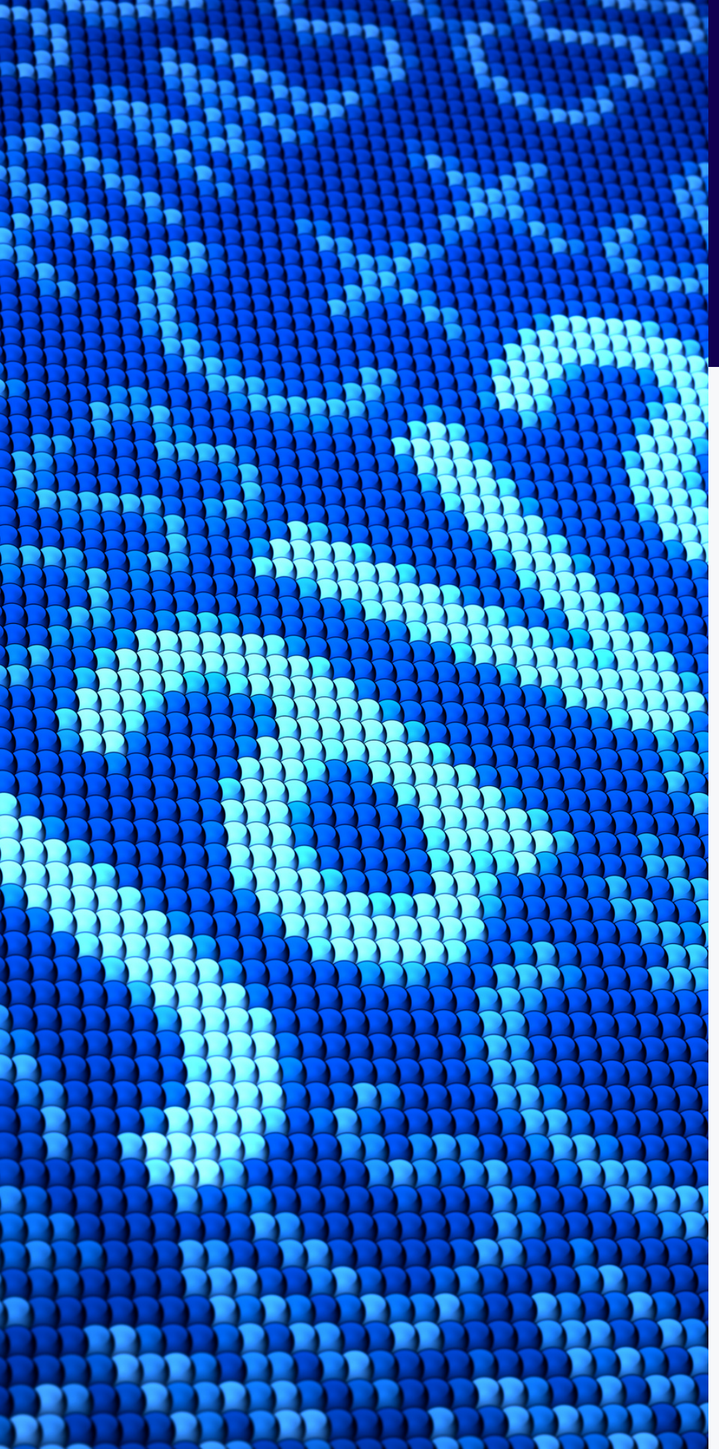
Hedef

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

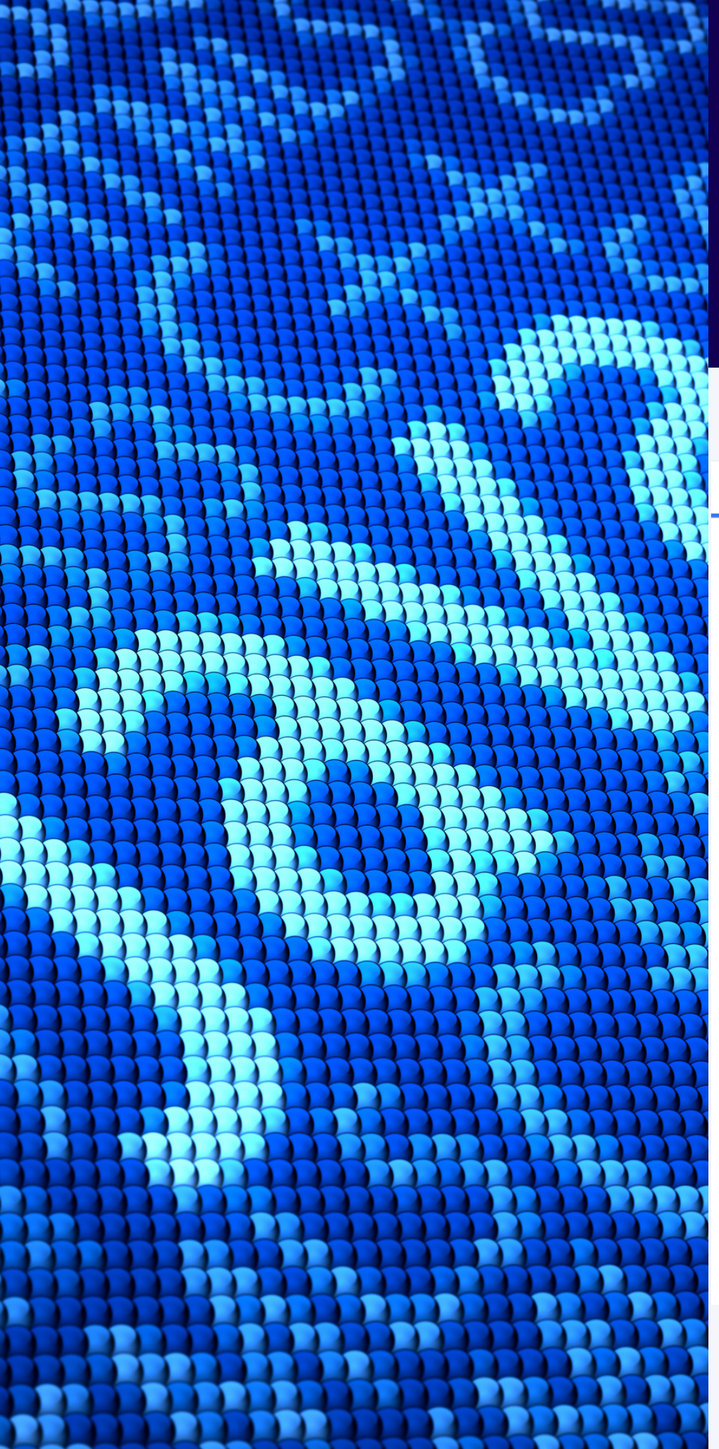
React - Front End Proje Yapısı



Spring Boot - Back End Proje Yapısı



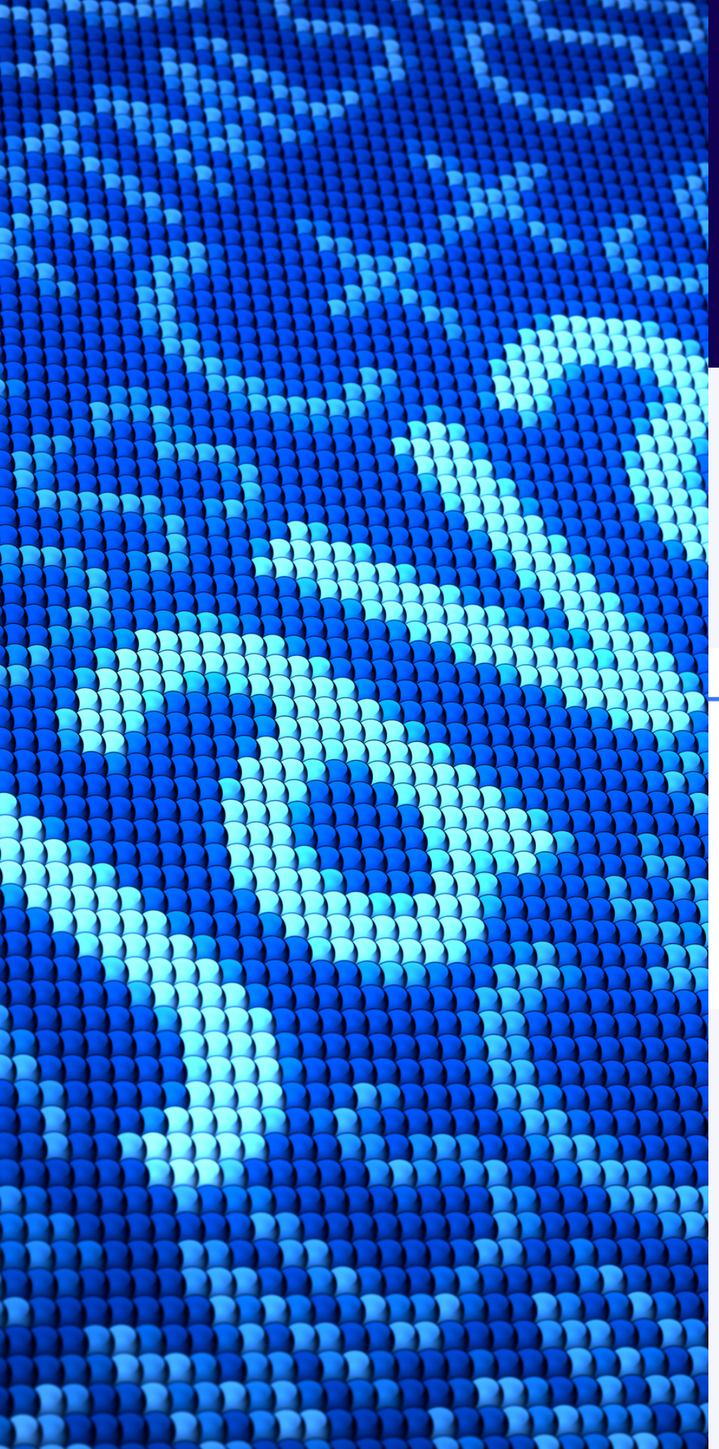
- src
  - main
    - java
      - com.iremnazyolcu
        - assist
          - FrontEnd
          - PostmanPersist
        - bean
          - ModelMapperBean
          - OpenApiConfigBean
          - PasswordEncoderBean
        - business
          - dto
            - ToDoDto
          - services
 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



Back End - ToDoEntity.java

ToDoEntity.java

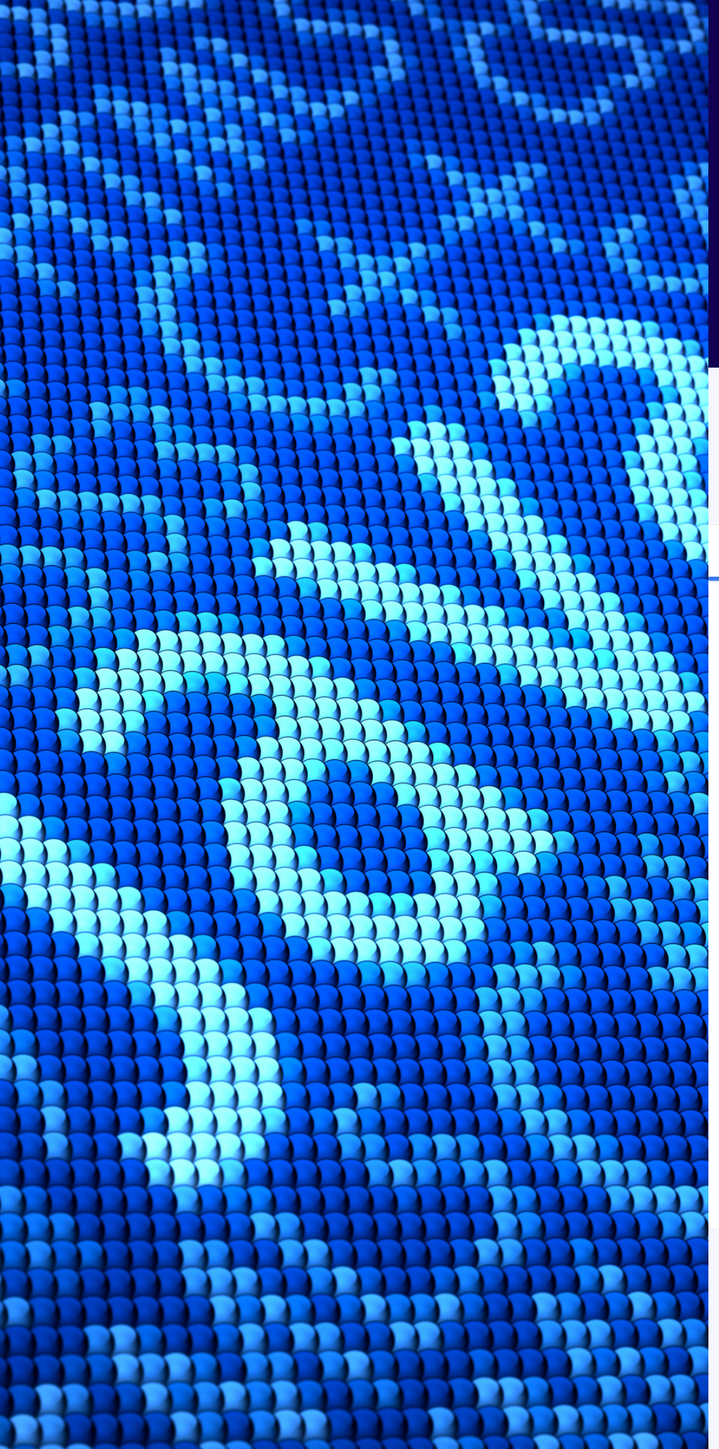
```java
package com.iremnazyolcu.data.entity;

import ...

// LOMBOK
21 usages
@Data

// ENTITY
@Entity
// Hibernate will create this table in a database with the following entities :
@Table(name = "todos")
public class ToDoEntity implements Serializable {

    // Serialization
    public static final Long serialVersionUID = 1L;

    // ID
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "todo_id", unique = true, nullable = false, insertable = true, updatable = false)
    private Long id;

    // TITLE
    @Column(nullable = false)
    private String title;

    // DESCRIPTION
    @Column(nullable = false)
    private String description;

    // COMPLETED
    private boolean completed;

```



Back End - ToDoRepository.java

ToDoRepository.java

```java
package com.iremnazyolcu.data.repository;

import ...

3 usages
@Repository
// ToDoRepository will be able to perform CRUD operations on the Entity specified in place of entityName.
// JpaRepository<entityName, primaryKeyType>
public interface ToDoRepository extends JpaRepository<ToDoEntity, Long> {
    2 usages
    List<ToDoEntity> findAllByCompleted(boolean completed);
}
```



Back End - ToDoDto.java

ToDoDto.java

```java
package com.iremnazyolcu.business.dto;

import ...

// LOMBOK
48 usages
@Data
public class ToDoDto implements Serializable {

    // Serialization
    public static final Long serialVersionUID = 1L;

    // ToDoDto attributes :

    // ID
    private Long id;

    // TITLE
    private String title;

    // DESCRIPTION
    private String description;

    // COMPLETED
    private boolean completed;

}
```

Back End - ToDoService.java

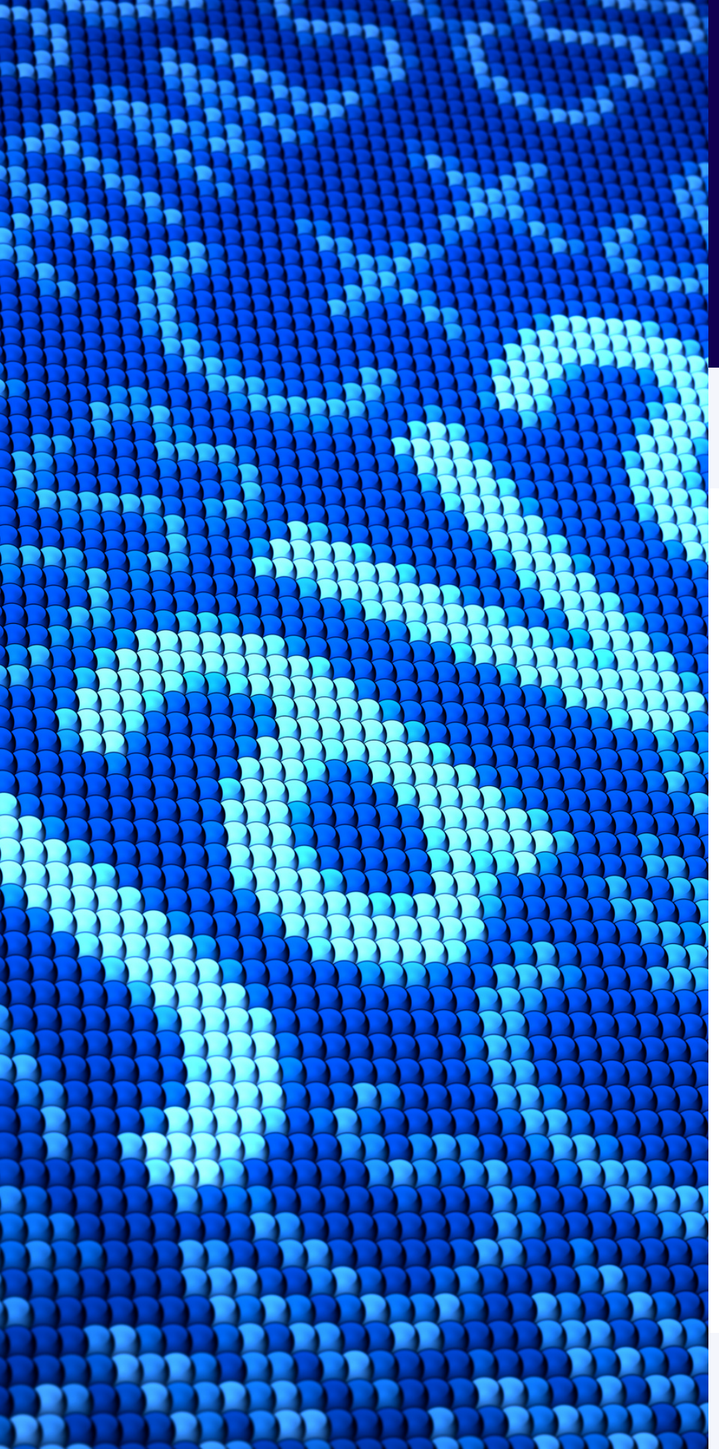
ToDoService.java

```java
// CREATE
1 usage  1 implementation
ToDoDto createTodo(ToDoDto todoDto);

// GET
1 usage  1 implementation
ToDoDto getTodo(Long id);

// GET ALL
2 usages  1 implementation
List<ToDoDto> getAllTodos();

// UPDATE
1 usage  1 implementation
ToDoDto updateTodo(ToDoDto todoDto, Long id);

// DELETE
1 usage  1 implementation
void deleteTodo(Long id);

// COMPLETE
1 usage  1 implementation
ToDoDto completeTodo(Long id);

// INCOMPLETE
1 usage  1 implementation
ToDoDto inCompleteTodo(Long id);

// DELETE ALL
1 usage  1 implementation
String deleteAllTodos();

// GET COMPLETED
1 usage  1 implementation
List<ToDoDto> getCompletedTodo(boolean completed);

// DELETE COMPLETED
1 usage  1 implementation
void deleteCompletedTodo();
```

Back End - ToDoServiceImpl.java

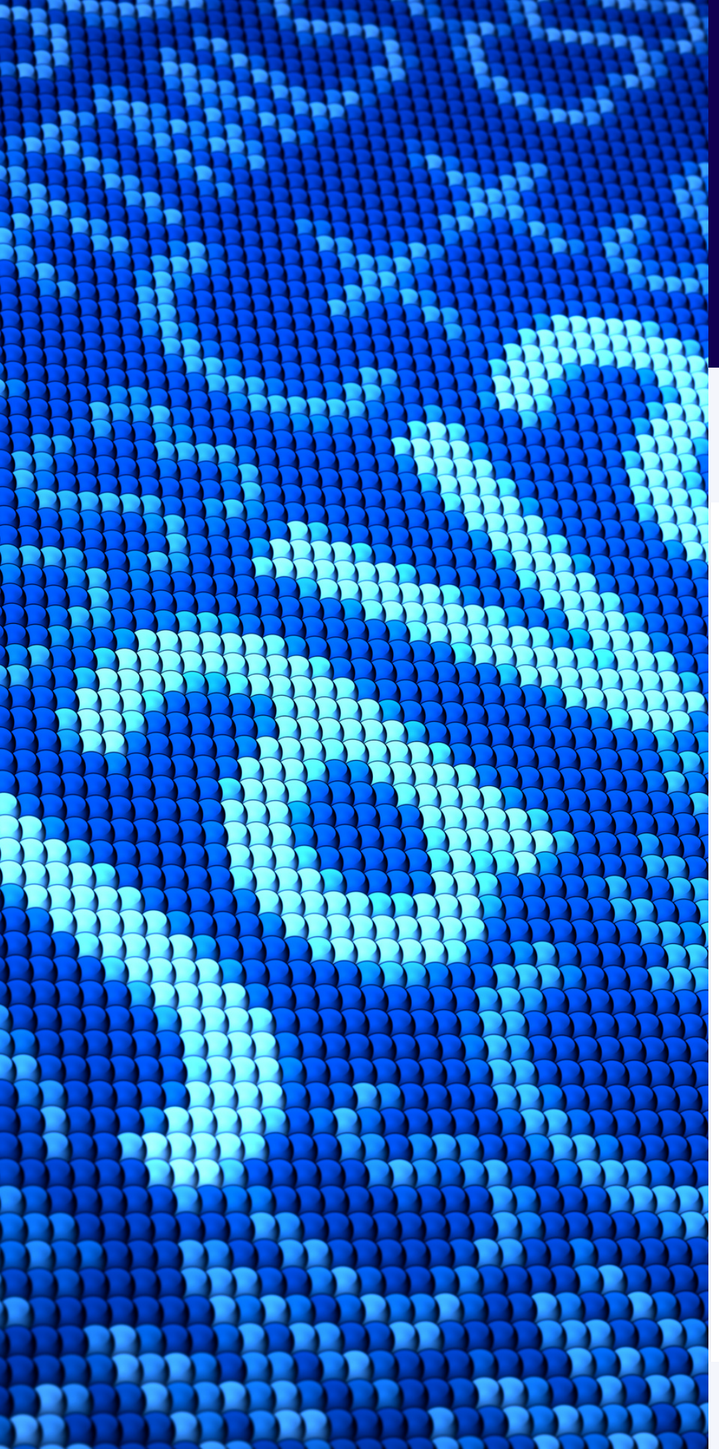
ToDoServiceImpl.java

```java

@Service
// Constructor based dependency injection
@AllArgsConstructor
// ToDoServiceImpl class implements the ToDoService interface and its methods.
public class ToDoServiceImpl implements ToDoService {

    // Injection
    // This ToDoServiceImpl requires ToDoRepository as a dependency.
    private ToDoRepository todoRepository;
    private ModelMapper modelMapper;

    // CREATE
    1 usage
    @Override
    public ToDoDto createTodo(ToDoDto todoDto) {

        // Model Mapper Library is one of the mapping library that we can use to automatically convert
        // DTO into Entity and Entity into DTO.
        ToDoEntity todo = modelMapper.map(todoDto, ToDoEntity.class);
        ToDoEntity savedTodo = todoRepository.save(todo);
        ToDoDto savedTodoDto = modelMapper.map(savedTodo, ToDoDto.class);
        return savedTodoDto;
    }

    // GET
    1 usage
    @Override
    public ToDoDto getTodo(Long id) {

        // retrieve the ToDo entity object from the database :
        ToDoEntity todo = todoRepository.findById(id)
                .orElseThrow(() -> new TodoNotFoundException("Todo not found with id:" + id));

        // getTodo() method return type is ToDoDto
        // so we should convert this ToDo entity object into ToDoDto :
        return modelMapper.map(todo, ToDoDto.class);

    }
```



Back End - ToDoServiceImpl.java

```java
// GET ALL
2 usages
@Override
public List<ToDoDto> getAllTodos() {
    List<ToDoEntity> todos = todoRepository.findAll();

    return todos.stream().map((todo) -> modelMapper.map(todo, ToDoDto.class))
            .collect(Collectors.toList());
}

// UPDATE
1 usage
@Override
public ToDoDto updateTodo(ToDoDto todoDto, Long id) {

    ToDoEntity todo = todoRepository.findById(id)
            .orElseThrow(() -> new TodoNotFoundException("Todo not found with id : " + id));
    todo.setTitle(todoDto.getTitle());
    todo.setDescription(todoDto.getDescription());
    todo.setCompleted(todoDto.isCompleted());

    ToDoEntity updatedTodo = todoRepository.save(todo);

    return modelMapper.map(updatedTodo, ToDoDto.class);
}

// DELETE
1 usage
@Override
public void deleteTodo(Long id) {

    ToDoEntity todo = todoRepository.findById(id)
            .orElseThrow(() -> new TodoNotFoundException("Todo not found with id : " + id));

    todoRepository.deleteById(id);
}
```

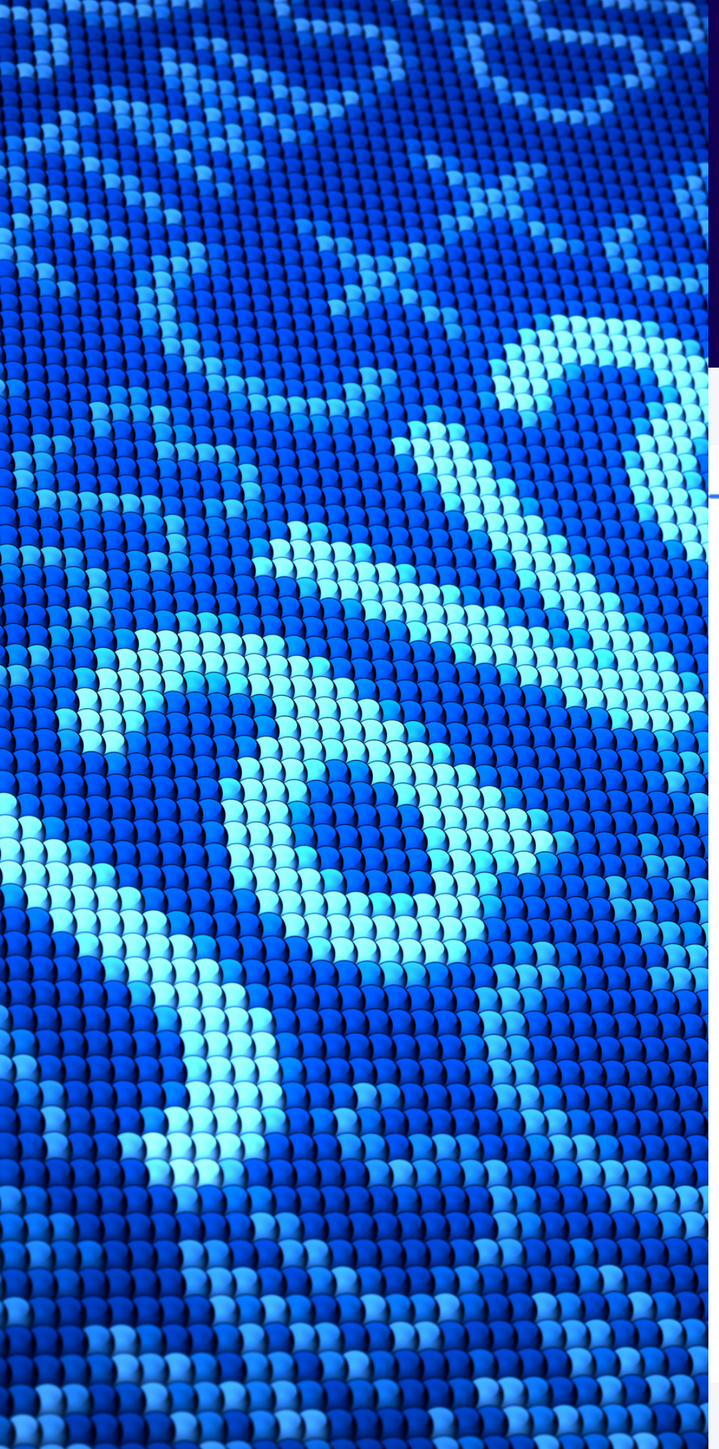


Back End - ToDoServiceImpl.java

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

Back End - ToDoServiceImpl.java

```java
    // GET COMPLETED
    1 usage
    @Override
    public List<ToDoDto> getCompletedTodo(boolean completed) {
        List<ToDoEntity> completedTodos = todoRepository.findAllByCompleted(completed);

        return completedTodos.stream() Stream<ToDoEntity>
                .map(entity -> modelMapper.map(entity, ToDoDto.class)) Stream<ToDoDto>
                .collect(Collectors.toList());

    }

    // DELETE COMPLETED
    1 usage
    @Override
    public void deleteCompletedTodo() {
        List<ToDoEntity> completedTodos = todoRepository.findAllByCompleted(true);
        for (ToDoEntity completedTodo : completedTodos) {
            todoRepository.delete(completedTodo);

        }

    }
```



Back End - ToDoController.java

ToDoController.java

```java
package com.iremnazyolcu.controller.api;

import ...

// The @RestController annotation designates this class as a Spring MVC REST controller.
// Within this Spring MVC REST controller, we can define the REST APIs.
@RestController
// @RequestMapping annotation establishes the base URL for all the REST APIs within the ToDoController class.
@RequestMapping("api/todos")
// Constructor based dependency injection
@AllArgsConstructor
@CrossOrigin(origins = FrontEnd.REACT_URL)
public class ToDoController {

    // Injection
    // This ToDoController requires ToDoService as a dependency.
    private ToDoService todoService;

    // Constructor based dependency injection

    // CREATE
    // http://localhost:4444/api/todos/create
    @PostMapping("/create")
    public ResponseEntity<ToDoDto> createTodo(@RequestBody ToDoDto todoDto) {
        ToDoDto createdTodo = todoService.createTodo(todoDto);
        return new ResponseEntity<>(createdTodo, HttpStatus.CREATED);
    }

    // GET
    // http://localhost:4444/api/todos/list/1
    @GetMapping("/list/{id}")
    public ResponseEntity<ToDoDto> getTodo(@PathVariable("id") Long id) {
        ToDoDto todo = todoService.getTodo(id);
        if (todo != null) {
            return new ResponseEntity<>(todo, HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
```

Back End - ToDoController.java

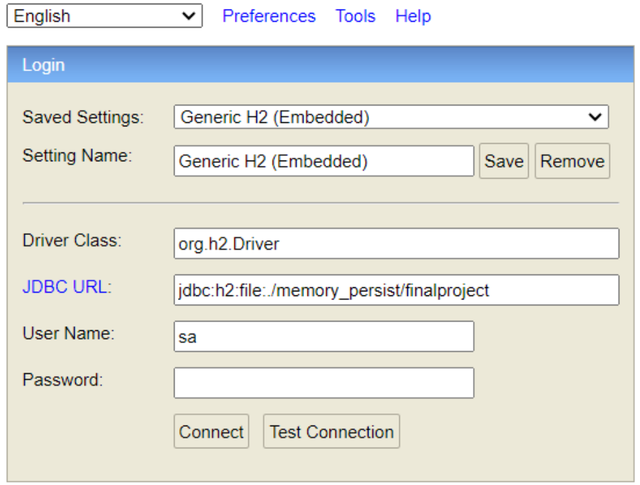
```java
// GET ALL
// http://localhost:4444/api/todos/list/all
@GetMapping("/list/all")
public ResponseEntity<List<ToDoDto>> getAllTodos() {
    List<ToDoDto> todos = todoService.getAllTodos();
    return new ResponseEntity<>(todos, HttpStatus.OK);
}

// UPDATE
// http://localhost:4444/api/todos/update/1
@PutMapping("/update/{id}")
public ResponseEntity<ToDoDto> updateTodo(@RequestBody ToDoDto todoDto, @PathVariable("id") Long id) {
    ToDoDto updatedTodo = todoService.updateTodo(todoDto, id);
    if (updatedTodo != null) {
        return new ResponseEntity<>(updatedTodo, HttpStatus.OK);
    } else {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}

// DELETE
// http://localhost:4444/api/todos/delete/1
@DeleteMapping("/delete/{id}")
public ResponseEntity<String> deleteTodo(@PathVariable("id") Long id) {
    todoService.deleteTodo(id);
    return ResponseEntity.ok( body: "Todo deleted successfully!");
}

// DELETE ALL
// http://localhost:4444/api/todos/delete/all
@DeleteMapping(value = "/delete/all")
public ResponseEntity<String> allDeleteTodo() { return ResponseEntity.ok(todoService.deleteAllTodos()); }
```

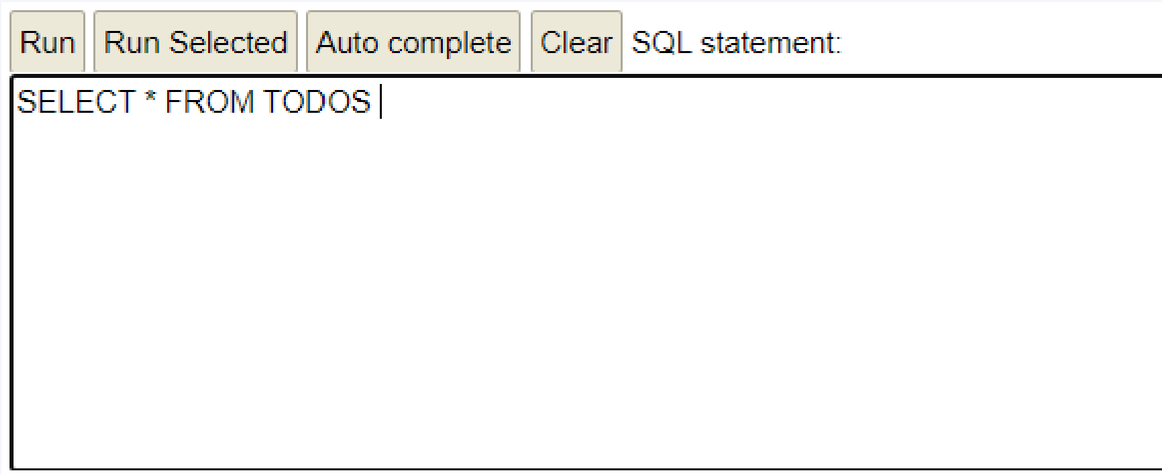
Back End - ToDoController.java

```java
    // COMPLETE
    // http://localhost:4444/api/todos/complete/1
    @PatchMapping("/complete/{id}")
    public ResponseEntity<ToDoDto> completeTodo(@PathVariable("id") Long id) {
        ToDoDto updatedTodo = todoService.completeTodo(id);
        return ResponseEntity.ok(updatedTodo);
    }

    // GET COMPLETED
    // http://localhost:4444/api/todos/completed/true
    @GetMapping("/completed/{completed}")
    public ResponseEntity<List<ToDoDto>> getCompletedTodos(@PathVariable boolean completed) {
        List<ToDoDto> completedTodos = todoService.getCompletedTodo(completed);
        return ResponseEntity.ok(completedTodos);
    }

    // DELETE COMPLETED
    // http://localhost:4444/api/todos/delete/completed
    @DeleteMapping("/delete/completed")
    public ResponseEntity<String> deleteCompletedTodo() {
        todoService.deleteCompletedTodo();
        return ResponseEntity.ok( body: "All completed todos deleted successfully!");
    }

    // INCOMPLETE
    // http://localhost:4444/api/todos/incomplete/1
    @PatchMapping("/incomplete/{id}")
    public ResponseEntity<ToDoDto> inCompleteTodo(@PathVariable("id") Long id) {
        ToDoDto updatedTodo = todoService.inCompleteTodo(id);
        return ResponseEntity.ok(updatedTodo);
    }
```

Database Yapısı

todos		
PK	id	Long
	title	String
	description	String
	completed	boolean



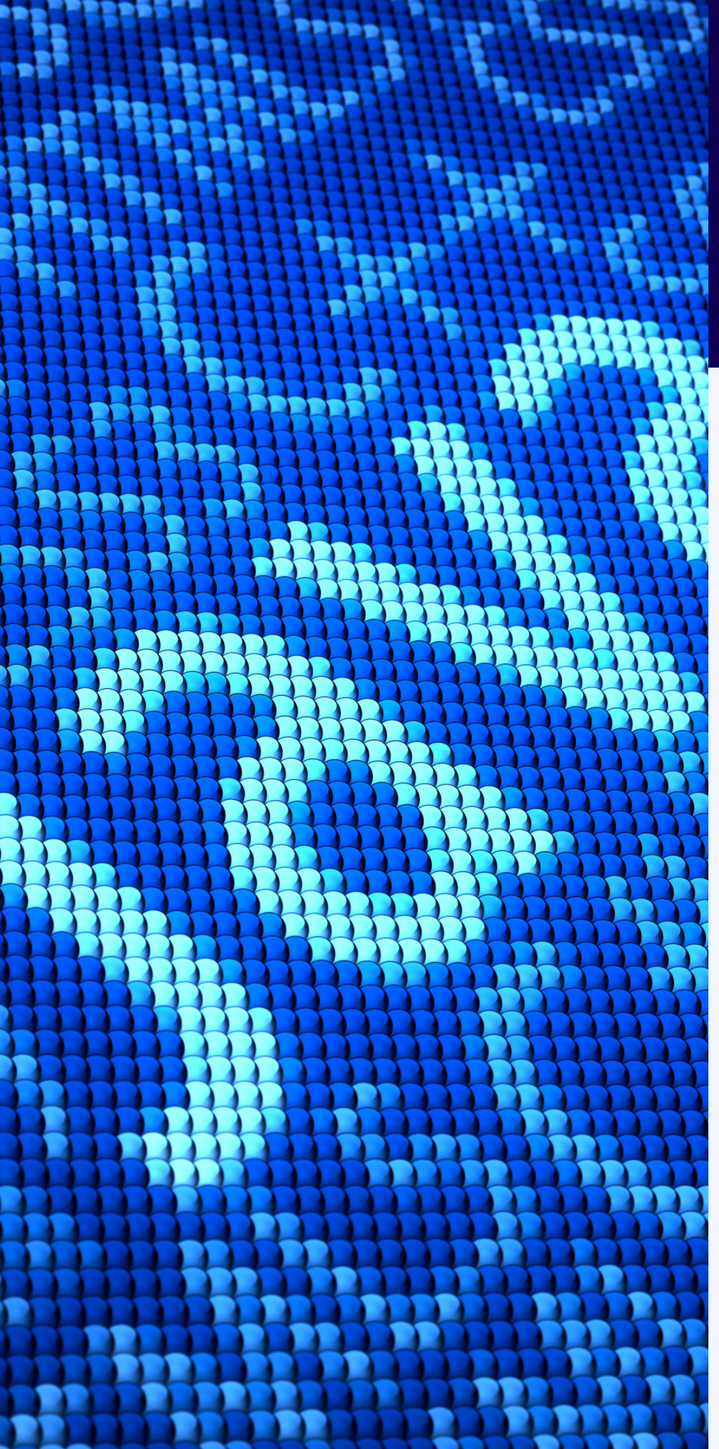


SELECT * FROM TODOS;

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

Add Todo

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

İremnaz Yolcu

iremnazyolcu@gmail.com

https://github.com/iremnazyolcuu

http://www.linkedin.com/in/iremnazyolcu

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