

TODO App Backend

Fixing a bug



Before

```
@PostMapping
fun putTodo(
    @RequestBody todoRequest: TodoRequest
) {
    todoService.createOrUpdate(todoRequest.toEntity())
}

data class TodoRequest(
    val task: String,
    val completed: Boolean = false
) {
    fun toEntity() = TodoEntity(
        task = task,
        completed = completed
    )
}
```

Bug: We always created a new instance of TodoEntity. Thereby we generate a new id every time and will not be able to update an entity

42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66

After

```
@PostMapping
fun putTodo(
    @RequestBody todoRequest: TodoRequest
) {
    val todo = todoRequest.id?.let {
        todoService.getById(it).apply {
            task = todoRequest.task
            completed = todoRequest.completed
        }
    } ?: todoRequest.toEntity()
    todoService.createOrUpdate(todo)
}

data class TodoRequest(
    val id: Int? = null,
    val task: String,
    val completed: Boolean = false
) {
    fun toEntity() = TodoEntity(
        task = task,
        completed = completed
    )
}
```

Fix: add optional ,id' to the request.
if id is part of the request we check
If this id exists, update the
corresponding data accordingly.
if id not exists we create a new
entity.

Adjust corresponding Tests

We are facing the bug because of a missing test.
It is super important to focus on tests as much as on the business logic.
It would have saved us from running into the bug.

TodoControllerTest.kt

```
@Test
fun `can add new todo`() {
    mockMvc.put(urlTemplate: "/todo") { this: MockHttpServletRequestDsl
        contentType = MediaType.APPLICATION_JSON
        content = TodoRequest(task = "test").asJsonString()
    }.andExpect { this: MockMvcResultMatchersDsl
        status { isCreated() }
    }
}
```

Before we only checked if we can create Todo.
This test is still valid and needed.

```
@Test
fun `can update todo`() {
    val anExistingTodoId : Int = todoRepository.saveAll(testData).random().id

    mockMvc.put(urlTemplate: "/todo") { this: MockHttpServletRequestDsl
        contentType = MediaType.APPLICATION_JSON
        content = TodoRequest(
            id = anExistingTodoId,
            task = "check if updating a task works"
        ).asJsonString()
    }.andExpect { this: MockMvcResultMatchersDsl
        status { isCreated() }
    }

    val updatedTodo : TodoEntity? = todoRepository.findByIdOrNull(anExistingTodoId)
    Assertions.assertThat(updatedTodo?.task).isEqualTo("check if updating a task works")
}
```

Let's add our missing test to prove we can update
existing todos via our rest endpoint.