Building the Web Layer

Esteban Herrera
Author

@eh3rrera | eherrera.net

Identified Endpoints

POST /tickets

PUT /tickets/{id}

GET /tickets/{id}

GET /tickets

Designing the Database Schema with TDD

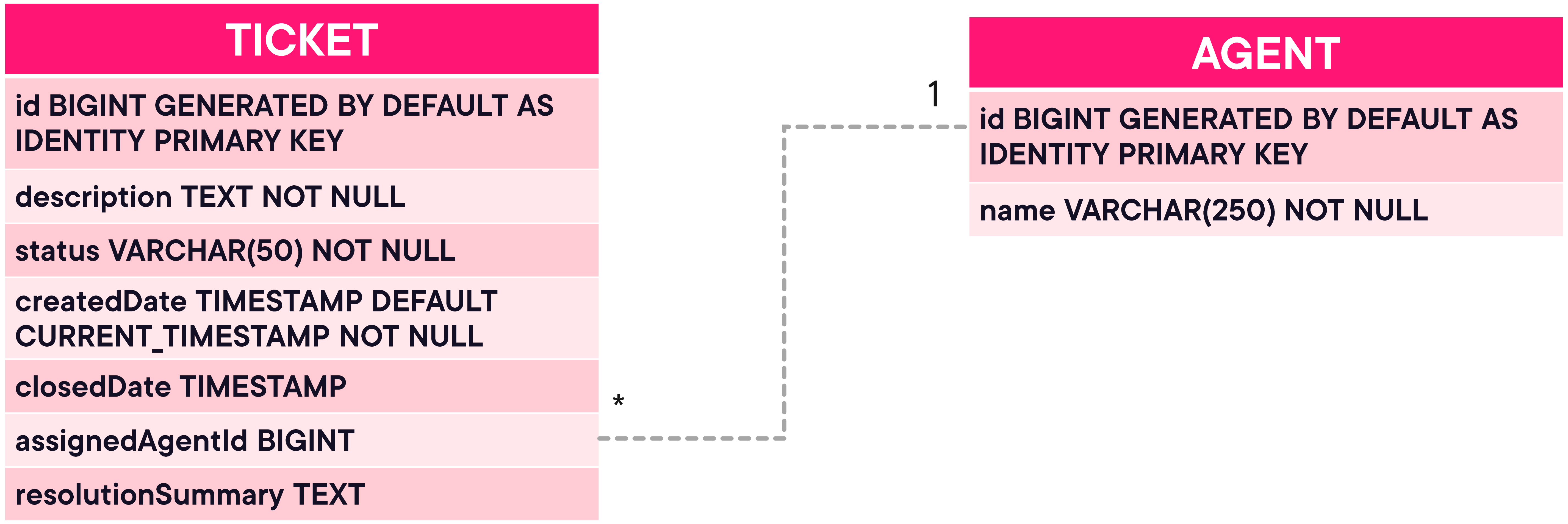
Just-In-Time

Upfront

Guidelines

- ✓ It's essential to remain flexible
- ✓ TDD’s iterative nature aligns with the just-in-time approach
- ✓ A hybrid approach often works well

Initial Database Schema


TICKET
id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY
description TEXT NOT NULL
status VARCHAR(50) NOT NULL
createdDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL
closedDate TIMESTAMP
assignedAgentId BIGINT
resolutionSummary TEXT
*
1
AGENT
id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY
name VARCHAR(250) NOT NULL

Validating Business Rules in the Service Layer

Pros	Cons
Separation of concerns	Complexity
Reusability	Initial development time
Localization of changes	Use of exceptions - Performance - Control flow

Exception Guidelines

- ✓ Be specific
- ✓ Don’t overuse
- ✓ Document exceptions
- ✓ Global exception handling

Creating the Project

Setting up the Test Class

Making Your Tests More Readable

AAA Pattern

Arrange

Act

Assert

FizzBuzz

```
@Test
void testNumberMultipleOfThree() {
    FizzBuzz fizzBuzz = new FizzBuzz();
    assertEquals("Fizz", fizzBuzz.convert(3));
}
```

Arrange

Assert

Act

FizzBuzz

```
@Test
void testNumberMultipleOfThree() {
    FizzBuzz fizzBuzz = new FizzBuzz();
    assertEquals("1", fizzBuzz.convert(1));
    assertEquals("2", fizzBuzz.convert(2));
}
```

MockMvc

```java
ResultActions resultActions =
    mockMvc.perform(MockMvcRequestBuilders.get("/some/endpoint"));

// Extract the MvcResult object for more detailed inspection
MvcResult mvcResult = resultActions.andReturn();
String responseBody = mvcResult.getResponse().getContentAsString();
```

MockMvc

```
ResultActions resultActions =
    mockMvc.perform(MockMvcRequestBuilders.get("/some/endpoint"));

// Print the response details
resultActions.andDo(MockMvcResultHandlers.print())
```

MockMvc

```
ResultActions resultActions =
    mockMvc.perform(MockMvcRequestBuilders.get("/some/endpoint"));

// Assert the response
resultActions
    .andExpect(MockMvcResultMatchers.status().isOk())
    .andExpect(MockMvcResultMatchers.content().contentType(
                                   MediaType.APPLICATION_JSON))
    .andExpect(MockMvcResultMatchers.jsonPath("$.key").value("value"))
    .andExpect(MockMvcResultMatchers.header().string(
                                   "Header-Name", "Header-Value"));
```

MockMvc

```
MockMvc mockMvc = //...  --------▶  Arrange

mockMvc.perform(get("/some/endpoint")) --------▶ Act
          .andExpect(status().isOk())
          .andExpect(content().contentType(MediaType.APPLICATION_JSON))
Assert    .andExpect(jsonPath("$.key").value("value"))
          .andExpect(header().string("Header-Name", "Header-Value"));
```

BDD Naming Convention

Given

When

Then

Given/When/Then

```java
@DisplayName(
   "Given a ticket with its details, when the ticket is created, then the
    ticket is saved"
)
@Test
void givenTicketDetails_whenTicketIsCreated_thenTicketIsSaved() {
    // ...
}
```

Creating a Ticket

Assigning an Agent to a Ticket

Assign Agent Functionality

Requirement:

A ticket can only move from 'New' to 'In Progress' when it is assigned to an agent.

Endpoint:

PUT /tickets/{id}/agent/{agentId}

Updating the Ticket Status

Resolve Ticket Functionality

Requirement:

When a ticket that is currently 'In Progress' is marked as resolved, its status should be changed to 'Resolved'.

Endpoint:

PUT /tickets/{id}/resolve

Close Ticket Functionality

Requirements:

Before closing a ticket, the required field "resolution summary" must be filled out.

Once a ticket is 'Closed', it cannot be reopened.

Endpoint:

PUT /tickets/{id}/close

Updating Tickets

Getting Tickets

Get Ticket Functionality

Requirement:

Get a ticket by its ID.

Endpoint:

GET /tickets/{id}

Get All Tickets Functionality

Requirement:

Get all tickets, optionally filtering by status, creation date range, and assigned agent.

Endpoint:

GET /tickets

Other Tests (For Error Scenarios)

Up Next:

Building the Service Layer