

Requirements:

Applying design-first approach, using the API design tool of your choice, please **design** a RESTful "Ticketing" API, obeying the following rules:

- Everyone can see the tickets
- Only authenticated users can sale or purchase tickets
- Only the ticket owner can modify the ticket (e.g. change price)

Implement using ASP.NET WebAPI, preferably in C#, only the update ticket endpoint described by your design and cover it with automated **tests**.

Solution:

- clean architecture
- web ui, api and testing startup projects
- **not all requirements implemented** (missing authentication, sale/purchase tickets, all users can modify tickets not just the owner, testing not complete)

Web app: <https://localhost:44383/>

API Design: <https://localhost:44363/swagger/index.html>

Owners

GET

/api/Owners

GET

/api/Owners/{id}

GET

/api/owners/{oid}/tickets

Tickets

GET

/api/Tickets

GET

/api/Tickets/{id}

PUT

/api/Tickets/{id}

<https://anileve.atlassian.net/secure/RapidBoard.jspa?rapidView=1&projectKey=RTA> (unfinished)

- Install .NET Core
- Install Visual Studio 2019
- Install Blazor
- using in-memory database

1. Everyone can see the tickets

Acceptance Criteria: Given the ticketing application, when any guest user accesses it, a list of all tickets is displayed.

Any user of the application can see the tickets both in the UI or using the API.

<https://localhost:44383/tickets>

2. Only authenticated users can sale or purchase tickets

"As an authenticated user, I want to sell/buy tickets."

Acceptance Criteria:

Given the user, when it is authenticated, it can buy/sell tickets

3. Only the ticket owner can modify the ticket (e.g. change price)

"As a ticket owner, I want to modify the ticket (e.g. change price)."

Acceptance Criteria:

Given an existing ticket, when it is selected by the owner, the ticket can be edited.

(partially implemented)

PUT /api/tickets/1{id}

Parameters

Name	Description
id	required
username	username
password	password

Request body

```
{
  "ticketId": 1,
  "username": "user",
  "password": "password",
  "price": 678
}
```

Response

```
{
  "ticketId": 1,
  "username": "user",
  "password": "password",
  "price": 678
}
```

Response code 200

localhost:44383/tickets/5

RANDOM stock learning news

WebApp

Ticket

- Theater ticket
- Some theater ticket
- 678

Title

Theater ticket

Description

Some theater ticket

price

678

Save View All Tickets

4. Implement automated tests for the update ticket endpoint (*partially implemented*)

```
Microsoft Visual Studio Debug Console

/////////////////////////////////
Reading owners...
Owner: Jane Doe
Owner: Jhon Doe
Owner: Kiddo Doe
/////////////////////////////////
Reading owner's tickets...
Owner: Jane Doe
    Ticket: Theater ticket
    Ticket: Gym ticket
    Ticket: Spa ticket
/////////////////////////////////
Reading all tickets..
Ticket: Some NEW title
Ticket: Theater ticket
Ticket: Opera ticket
Ticket: Theater ticket
Ticket: Theater ticket
Ticket: Gym ticket
Ticket: Spa ticket
Ticket: Gym ticket
Ticket: Opera ticket
Ticket: Spa ticket
/////////////////////////////////
Reading ticket id 1...
Ticket: Some NEW title
Ticket: Theater ticket
Ticket: Opera ticket
Ticket: Theater ticket
Ticket: Theater ticket
```