Meilenstein 2
Warehouse API

Rick Pleeging, Ibrahim Hussein und Hannes Prochaska

Meilenstein 2

Fertigstellung der Kommunikation zwischen API und Datenbank, Unit-Tests	02.04.2025

Fertigstellung der Kommunikation zwischen API und Datenbank

- RDBMS => postgreSQL
- Daten => Brickset (JSON-Format)
- Ziel = Automatische Erstellung der Datenbank durch C#-Programm

Fertigstellung der Kommunikation zwischen API und Datenbank

Datenbeisspiel:


```
{
  "Sets": [
    {
      "Number": "001",
      "NumberVariant": 1,
      "Name": "Gears",
      "Year": 1965,
      "Theme": "Samsonite",
      "ThemeGroup": "Vintage",
      "Subtheme": "Basic Set",
      "Category": "Normal",
      "Released": true,
      "Pieces": 43,
      "Minifigs": 0,
      "Image": {
        "ThumbnailURL": "https://images.brickset.com/sets/small/001-1.jpg",
        "ImageURL": "https://images.brickset.com/sets/images/001-1.jpg"
      },
      "LEGOCom": {
        "US": {
          "RetailPrice": null,
          "DateFirstAvailable": null,
          "DateLastAvailable": null
        },
        "UK": {
          "RetailPrice": null,
          "DateFirstAvailable": null,
          "DateLastAvailable": null
        },
        "CA": {
          "RetailPrice": null,
          "DateFirstAvailable": null,
          "DateLastAvailable": null
        },
        "DE": {
          "RetailPrice": null,
          "DateFirstAvailable": null,
          "DateLastAvailable": null
        }
      },
      "PackagingType": "Box",
      "Availability": "Retail",
      "InstructionsCount": 0,
      "AgeRange": {
        "Min": 5,
        "Max": 12
      },
      "Dimensions": {
        "Height": "20.30",
        "Width": "39.50",
        "Depth": "5.10",
        "Weight": "0.00"
      },
      "Barcode": {
        "EAN": null,
        "UPC": null
      },
      "ExtendedData": {
        "BrickTags": []
      }
    },
```

Fertigstellung der Kommunikation zwischen API und Datenbank

- Idee => Lesen des JSON-Files und dynamische Erstellungen der Attribute (Modelle für EntitiyFramwork)
- Zusatz von NuGet-Packages zum arbeiten mit postgreSQL
- Überprüfung auf eigener postgreSQL-Datenbank
- pgAdmin 4 und Dbeaver zur grafischen Überschaubarkeit

Fertigstellung der Kommunikation zwischen API und Datenbank

In-Person-Review

Login testen

```
0 Verweise
public void Login_WithValidCredentials()
{
    // Einfachen InMemory-Datenbank-Kontext erstellen
    var options = new DbContextOptionsBuilder<ApplicatonDbContext>()
        .UseInMemoryDatabase(databaseName: "TestDb")
        .Options;
    var context = new ApplicatonDbContext(options);

    // Benutzer in die Datenbank einfügen
    var user = new Users { email = "test@example.com", HashedPassword = BCrypt.Net.BCrypt.HashPassw
    context.Users.Add(user);
    context.SaveChanges();

    // Controller erstellen
    var config = new ConfigurationBuilder().Build();
    var controller = new LoginController(context, new AuthService(config), new ApiKey(config));

    // Login-Daten vorbereiten
    var loginRequest = new Users { email = "test@example.com", HashedPassword = "123456" };

    // Login aufrufen
    var result = controller.Login(loginRequest) as OkObjectResult;

    // Überprüfen
    Assert.NotNull(result);
    Assert.Equal(200, result!.StatusCode);
}
```

Login Testen

```
public void Login_WithInvalidPassword()
{
    // InMemory-Datenbank vorbereiten
    var options = new DbContextOptionsBuilder<ApplicatonDbContext>()
        .UseInMemoryDatabase("TestDb_LoginFail")
        .Options;
    var context = new ApplicatonDbContext(options);

    // Benutzer speichern
    var user = new Users { email = "test@example.com", HashedPassword = BCrypt.Net.BCrypt.HashPassword("123456") };
    context.Users.Add(user);
    context.SaveChanges();

    // Controller erstellen
    var config = new ConfigurationBuilder().Build();
    var controller = new LoginController(context, new AuthService(config), new ApiKey(config));

    // Falsches Passwort beim Login
    var loginRequest = new Users { email = "test@example.com", HashedPassword = "wrongpassword" };

    // Login aufrufen
    var result = controller.Login(loginRequest) as UnauthorizedObjectResult;

    // Überprüfen
    Assert.NotNull(result);
    Assert.Equal(401, result!.StatusCode);
}
```

Allê ARÍ ķêỳș ăčsụğêŋ

```
[Fact]
0 Verweise
public void GetAllApiKeys_ReturnsKeys()
{
    // Arrange
    var options = new DbContextOptionsBuilder<ApplicatonDbContext>()
        .UseInMemoryDatabase("GetAllApiKeys")
        .Options;
    var context = new ApplicatonDbContext(options);

    var config = new ConfigurationBuilder().Build();
    var apiKeyService = new ApiKey(config);
    var controller = new LoginController(context, new AuthService(config), apiKeyService);

    // Generiere API-Key
    controller.GenerateApiKey();

    // Act
    var result = controller.GetAllApiKeys() as OkObjectResult;

    // Assert
    Assert.NotNull(result);
}
```

- Admin Key mit gültigen API Key testen

```
public void ProtectedEndpoint_WithValidApiKey_ReturnsOk()
{
    // Arrange
    var options = new DbContextOptionsBuilder<ApplicatonDbContext>()
        .UseInMemoryDatabase("ProtectedOk")
        .Options;
    var context = new ApplicatonDbContext(options);

    var config = new ConfigurationBuilder().Build();
    var apiKeyService = new ApiKey(config);
    var controller = new LoginController(context, new AuthService(config), apiKeyService);

    // Holen des Admin-API-Schlüssels
    var adminKey = apiKeyService.GetAdminKey();

    // Act
    var result = controller.GetProtectedData(adminKey) as OkObjectResult;

    // Assert
    Assert.NotNull(result);
    Assert.Equal(200, result.StatusCode);  // Überprüft, dass der Statuscode 200 (OK) ist
}
```

- Ungültie Admin Key mit gültigen API Key testen

```
0 Verweise
public void ProtectedEndpoint_WithInvalidApiKey_ReturnsUnauthorized()
{
    // Arrange
    var options = new DbContextOptionsBuilder<ApplicatonDbContext>()
        .UseInMemoryDatabase("ProtectedFail")
        .Options;
    var context = new ApplicatonDbContext(options);

    var config = new ConfigurationBuilder().Build();
    var apiKeyService = new ApiKey(config);
    var controller = new LoginController(context, new AuthService(config), apiKeyService);

    // Act
    var result = controller.GetProtectedData("falscherKey") as UnauthorizedObjectResult;

    // Assert
    Assert.NotNull(result);
    Assert.Equal(401, result.StatusCode);
}
```