Project No. 1:
A Web application with database access

Szymon Górski

1 Project Setup

For creating my Web application, I used C# language, ASP.NET Core with Razor pages and Entity Framework (EF) Core technologies with Visual Studio IDE. All initial tables and data were inserted to a local database `TaskNo1` via `SQLQuery` query. The EF allows to import tables to an ASP.NET project using ADO.NET Entity Data Model and Entity Data Model Wizard. Unfortunately, setting this Model via Wizard is at least non-trivial, or maybe even impossible in ASP.NET Core, therefore after a few unsuccessful attempts I decided to add models of tables manually (`MOVIES.cs` and `ORDERS.cs` files). To bind a project with the database, I created data binding, which points to a database in `appsettings.json` file and points to tables in `MySolutionContext.cs` file.

2 SQL and C# similarities

The main Web page is `Index.cshtml`. Razor pages of movies and orders are placed in corresponding folders. `Movies/Index.cshtml.cs` holds the core of filtering movies. Methods have similar structure as in SQL language and use `Where` statement:

```
if (!String.IsNullOrEmpty(titleValue)) moviesIQ = moviesIQ.Where
        (s => s.MOVIE_TITLE.Contains(titleValue));
if (!String.IsNullOrEmpty(genreValue)) moviesIQ = moviesIQ.Where
        (s => s.GENRE.Contains(genreValue));
if (Convert.ToDecimal(ratingValue) > 0) moviesIQ = moviesIQ.Where
        (s => s.RATING >= Convert.ToDecimal(ratingValue));
if (fromValue.Year > 1800 && fromValue.Year < 2100) moviesIQ =
        moviesIQ.Where(s => s.RELEASE_DATE >= fromValue);
if (toValue.Year > 1800 && toValue.Year < 2100)
        moviesIQ = moviesIQ.Where(s => s.RELEASE_DATE <= toValue);
```

Creation of a new order (as specified in task in point 5c) is done in `Orders/Create.cshtml.cs`:

```
ORDERS = new ORDERS();
ORDERS.MOVIE_ID = id;
ORDERS.MOVIES = _context.MOVIES.Find(ORDERS.MOVIE_ID);
ORDERS.RENTAL_DATE = DateTime.Now;
if (DateTime.Now.AddYears(-5) >= ORDERS.MOVIES.RELEASE_DATE)
{
    ORDERS.RETURN_DATE = DateTime.Now.AddDays(7);
    ORDERS.DISCOUNT = discount;
    ORDERS.GROSS_AMOUNT = ORDERS.MOVIES.PRICE *
            (1M - discount) * 1.23M;
}
else
{
    ORDERS.RETURN_DATE = DateTime.Now.AddDays(3);
    ORDERS.DISCOUNT = 0M;
    ORDERS.GROSS_AMOUNT = ORDERS.MOVIES.PRICE * 1.23M;

}
ORDERS.NET_AMOUNT = ORDERS.MOVIES.PRICE;
_context.ORDERS.Add(ORDERS);
_context.SaveChanges();
```

3 Important details

Among many technology-specific solutions, which I needed to understand to finish the project, I found two of them especially important:

1. Creating IDs for rows by models works for out-of-the-box templates. However, after editing them, the mechanism might stop working. To solve it, it is necessary to specify that IDs are inserted using `IDENTITY(1, 1)`.

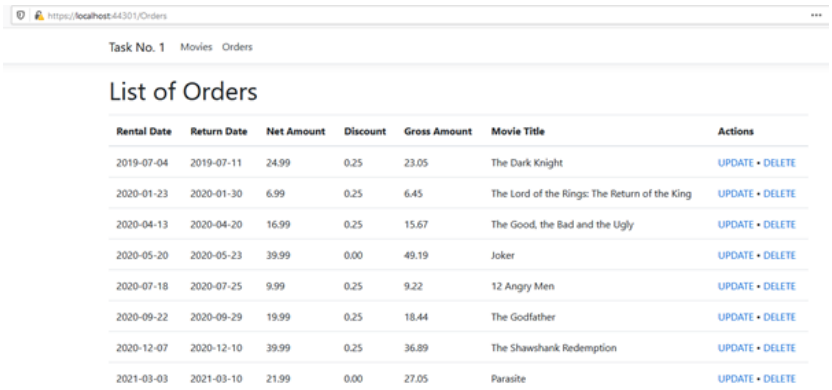
2. To prevent concurrency issues, it is enough to apply proper flags or objects in the models of tables. ASP.NET automatises conflicts in the optimistic way, which does not block edition, rather creates queue of editions and if data from query change while executing it, refuses to continue. This is applied via `[ConcurrencyCheck]` to columns which can be edited through my application. Right now the application does not implement this functionality.

4 Execution and screenshots

It is impossible to run the project immediately as ASP.NET settings point at the database stored on a local computer. It is necessary to load the database

from `SQLQuery.sql` and modify `appsettings.json` file first. Screenshots from the solution:

Initial `ORDERS` table:



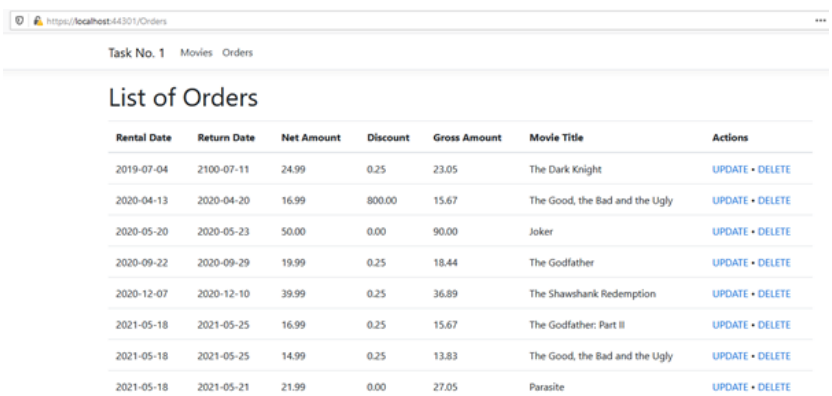
https://localhost:44301/Orders

Task No. 1 Movies Orders

List of Orders

Rental Date	Return Date	Net Amount	Discount	Gross Amount	Movie Title	Actions
2019-07-04	2019-07-11	24.99	0.25	23.05	The Dark Knight	UPDATE • DELETE
2020-01-23	2020-01-30	6.99	0.25	6.45	The Lord of the Rings: The Return of the King	UPDATE • DELETE
2020-04-13	2020-04-20	16.99	0.25	15.67	The Good, the Bad and the Ugly	UPDATE • DELETE
2020-05-20	2020-05-23	39.99	0.00	49.19	Joker	UPDATE • DELETE
2020-07-18	2020-07-25	9.99	0.25	9.22	12 Angry Men	UPDATE • DELETE
2020-09-22	2020-09-29	19.99	0.25	18.44	The Godfather	UPDATE • DELETE
2020-12-07	2020-12-10	39.99	0.25	36.89	The Shawshank Redemption	UPDATE • DELETE
2021-03-03	2021-03-10	21.99	0.00	27.05	Parasite	UPDATE • DELETE

ORDERS after applying 3 `RENT`s, 3 `UPDATE`s (first three rows) and 3 `DELETE`s:



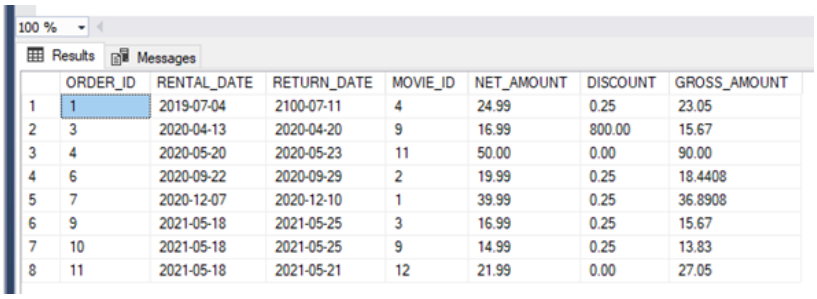
https://localhost:44301/Orders

Task No. 1 Movies Orders

List of Orders

Rental Date	Return Date	Net Amount	Discount	Gross Amount	Movie Title	Actions
2019-07-04	2100-07-11	24.99	0.25	23.05	The Dark Knight	UPDATE • DELETE
2020-04-13	2020-04-20	16.99	800.00	15.67	The Good, the Bad and the Ugly	UPDATE • DELETE
2020-05-20	2020-05-23	50.00	0.00	90.00	Joker	UPDATE • DELETE
2020-09-22	2020-09-29	19.99	0.25	18.44	The Godfather	UPDATE • DELETE
2020-12-07	2020-12-10	39.99	0.25	36.89	The Shawshank Redemption	UPDATE • DELETE
2021-05-18	2021-05-25	16.99	0.25	15.67	The Godfather: Part II	UPDATE • DELETE
2021-05-18	2021-05-25	14.99	0.25	13.83	The Good, the Bad and the Ugly	UPDATE • DELETE
2021-05-18	2021-05-21	21.99	0.00	27.05	Parasite	UPDATE • DELETE

Changes are visible via MS SQL as well:



100 %

Results Messages

	ORDER_ID	RENTAL_DATE	RETURN_DATE	MOVIE_ID	NET_AMOUNT	DISCOUNT	GROSS_AMOUNT
1	1	2019-07-04	2100-07-11	4	24.99	0.25	23.05
2	3	2020-04-13	2020-04-20	9	16.99	800.00	15.67
3	4	2020-05-20	2020-05-23	11	50.00	0.00	90.00
4	6	2020-09-22	2020-09-29	2	19.99	0.25	18.4408
5	7	2020-12-07	2020-12-10	1	39.99	0.25	36.8908
6	9	2021-05-18	2021-05-25	3	16.99	0.25	15.67
7	10	2021-05-18	2021-05-25	9	14.99	0.25	13.83
8	11	2021-05-18	2021-05-21	12	21.99	0.00	27.05