Game-club-admin

Vision

“Rental cars” is a web-application which allows admin to record information about customers, platforms and the orders.

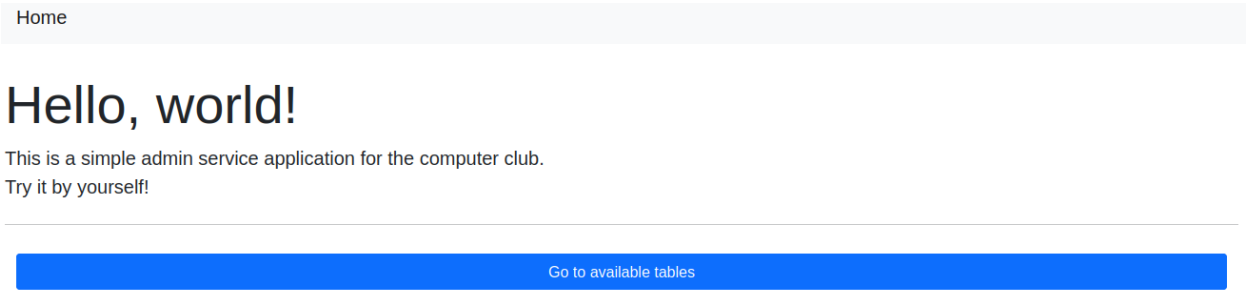
Application should provide:

- Storing gaming platform rent orders, customers logins and platforms by itself
- Display list of orders, customers and platforms
- Updating list of orders, platforms, customers (removing, adding or editing)
- Retrieving response of expired orders on the main page

Pages overview and tables relations

1. Main page

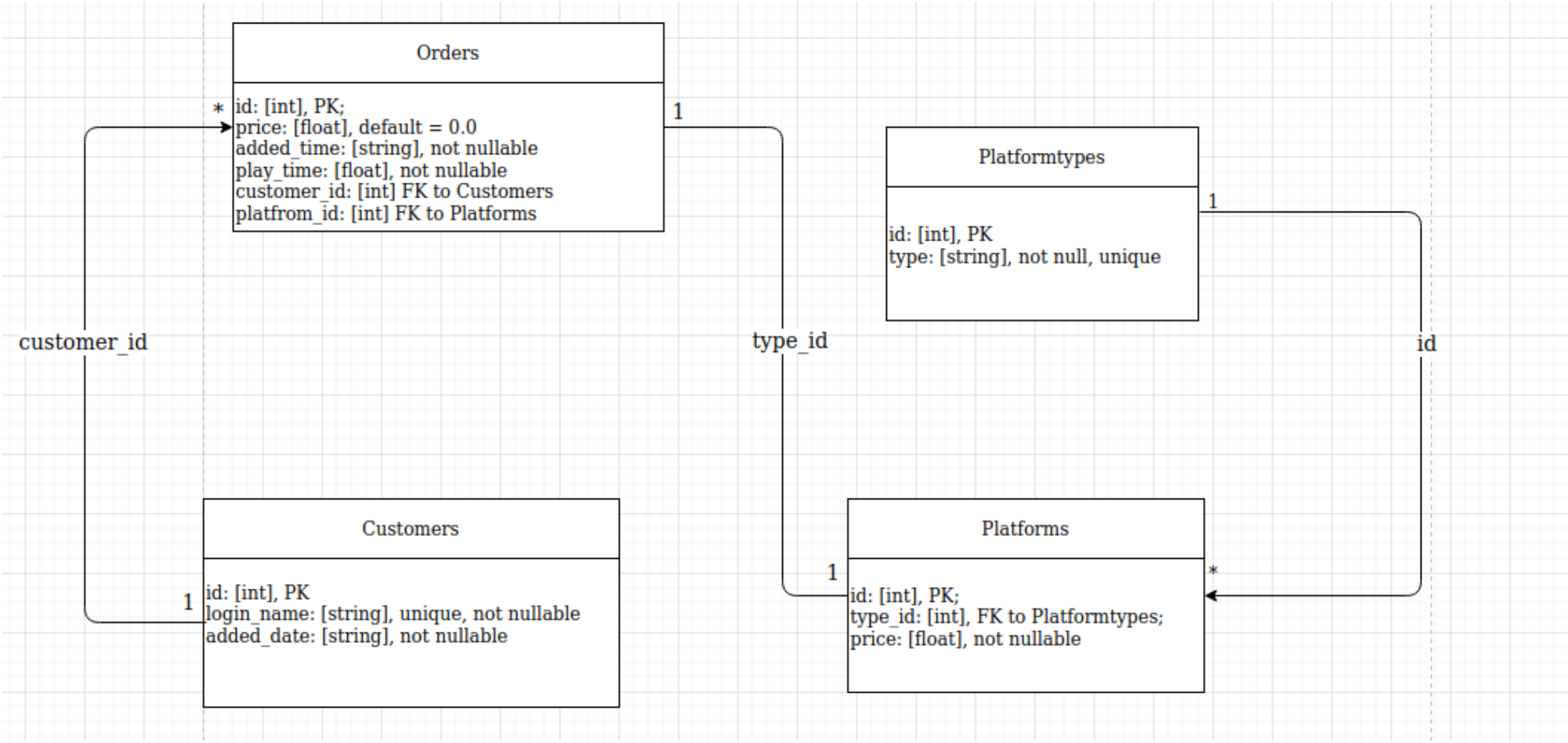
Welcome page gives to user a brief information about service and see the data in database with redirecting button



Pic. 1 Main page view

Later, on this page will be displayed information about game platform rent orders that expired by time.

For now, let’s see relations between tables in the service database.

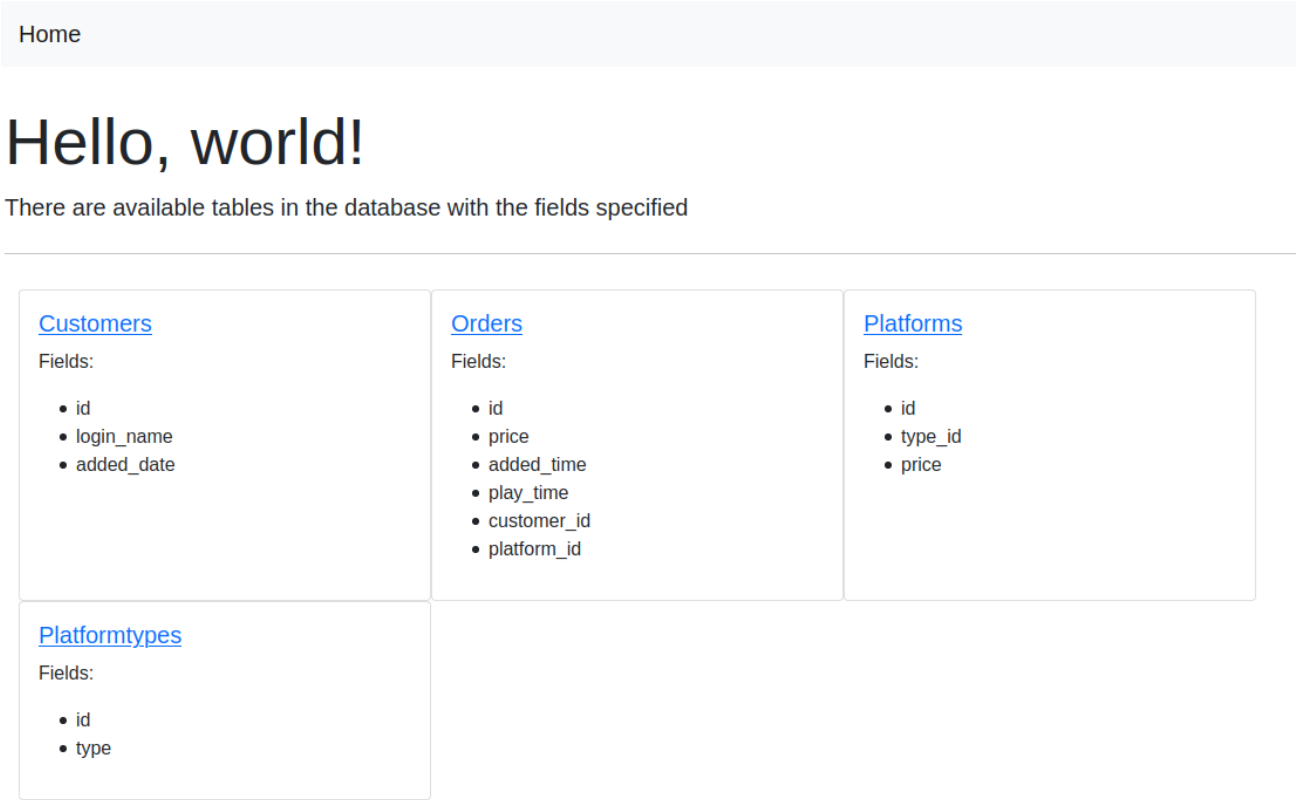


Pic. 2 Table relation UML

It can be seen that tables have add, edit and delete operation constraints upon rows, due to the relation between them. The exception is table `Platformtypes` only.

2. Available tables

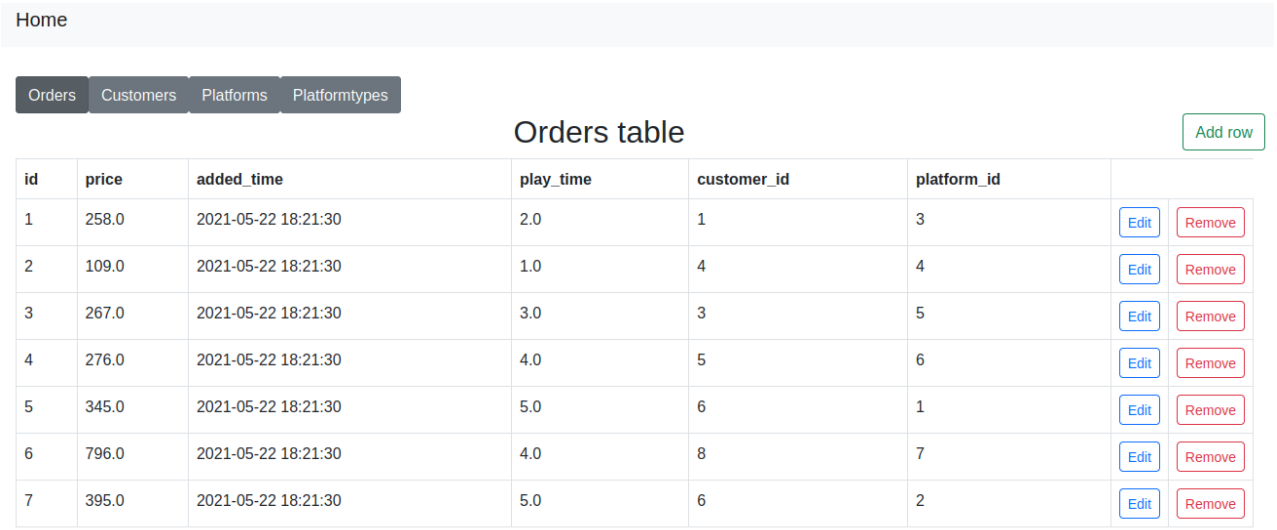
Available tables page consists of 4 short info cards about each table used in service.
Clicking on the specified table will redirect to its data representation page.



Pic. 2.1 Availables tables presentation view

Every card name has a link to the table show page.
Table pages, in turn, have similar interface and functionality.
As it can be seen, every table presentation view page has a navigation bar, to move between tables fast and easy.

Of course, the user is able to add new rows to a specific table, edit already presented data or remove rows.
Home button returns the user to the main welcome page of service.



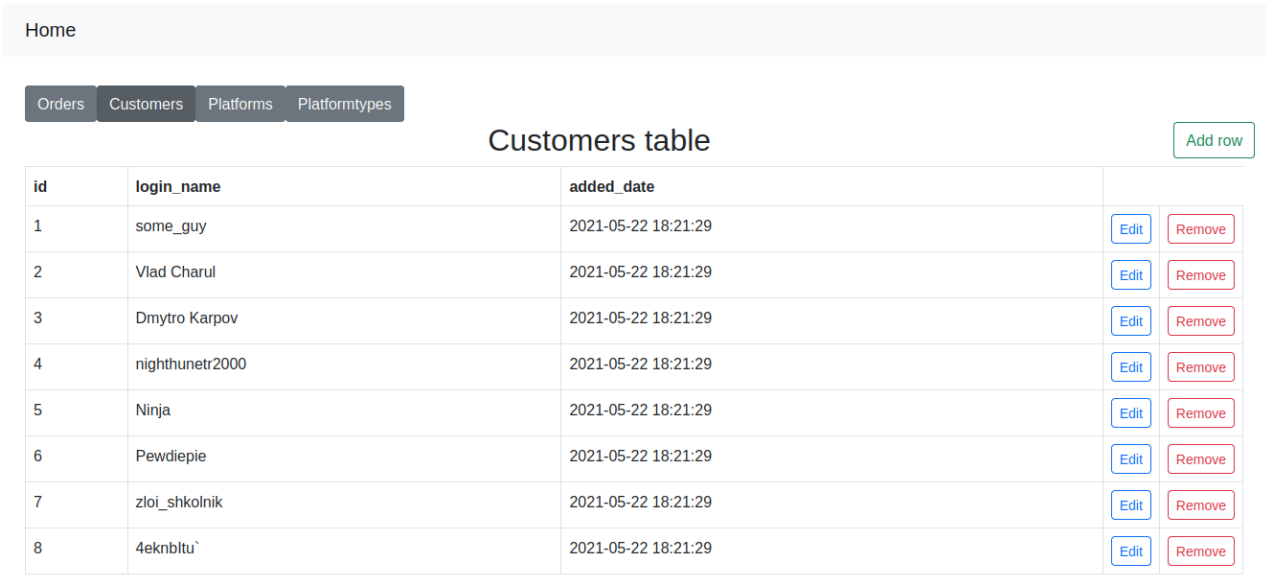
Pic. 2.2 Orders table presentation view

Fields, data from that inserts automatically into new row for table `Orders`:

- `id`
- `price`
- `added_time`

Fields, data from that inserts manually by user into new row for table `Orders`:

- `play_time`
- `customer_id`
- `platform_id`



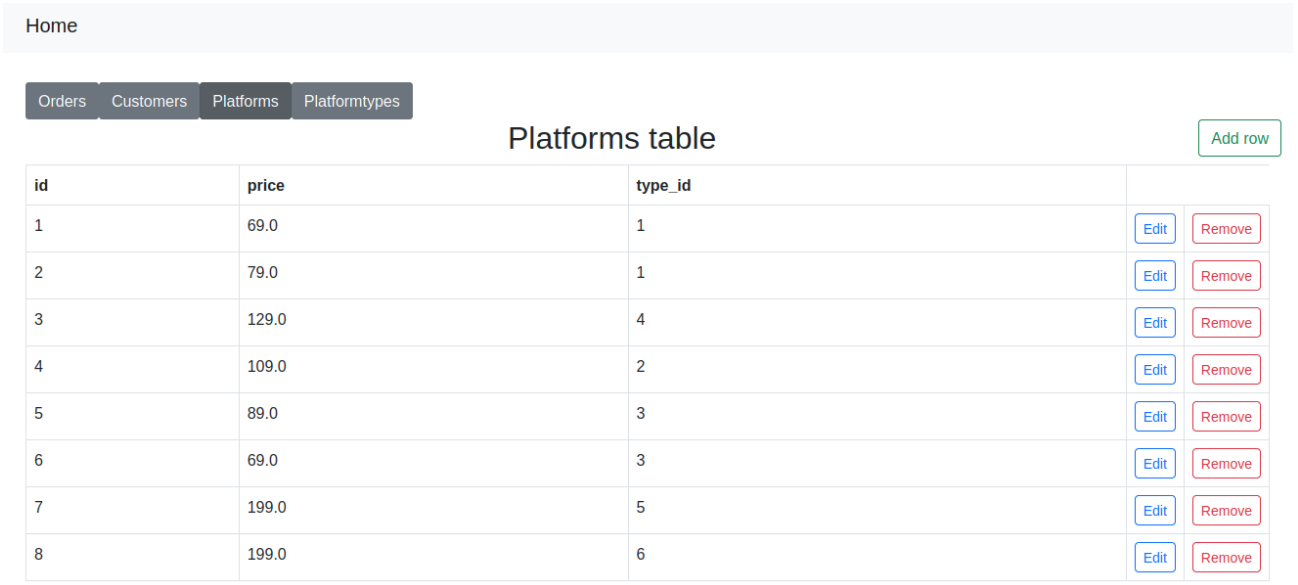
Pic. 2.3 Customers table presentation view

Fields, data from that inserts automatically into new row for table `Customers`:

- `id`
- `added_time`

Fields, data from that inserts manually by user into new row for table `Customers`:

- `login_name`



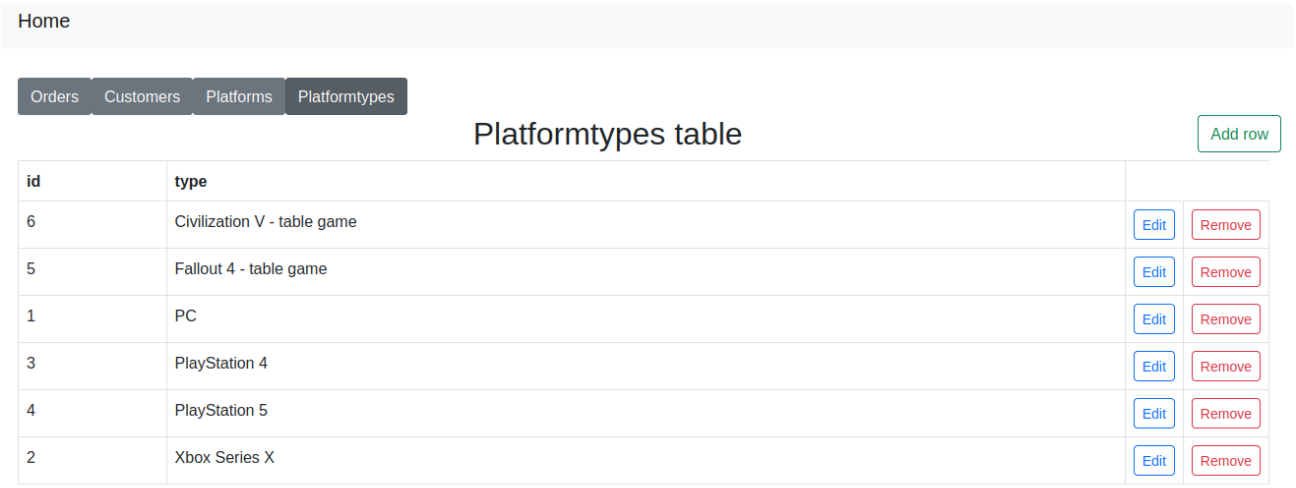
Pic. 2.4 Platforms presentation view

Fields, data from that inserts automatically into new row for table `Platforms`:

- `id`

Fields, data from that inserts manually by user into new row for table `Platforms`:

- `price`
- type_id



Pic. 2.5 Platformtypes presentation view

Fields, data from that inserts automatically into new row for table `Platforms`:

- `id`

Fields, data from that inserts manually by user into new row for table `Platforms`:

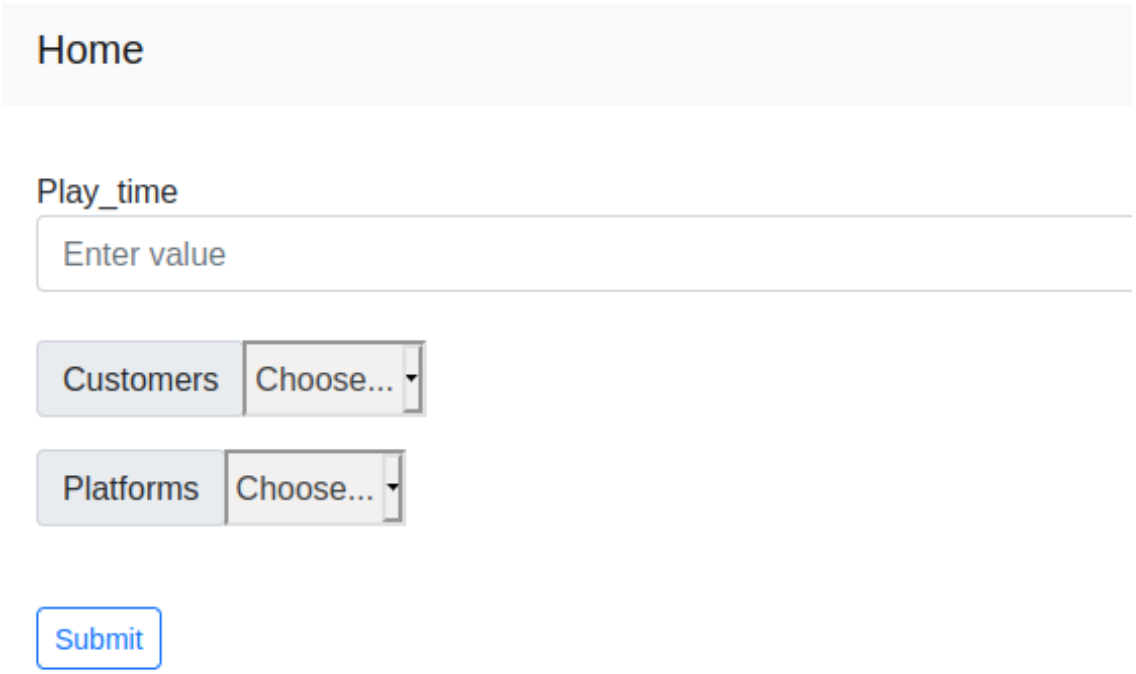
- `type`

Add data

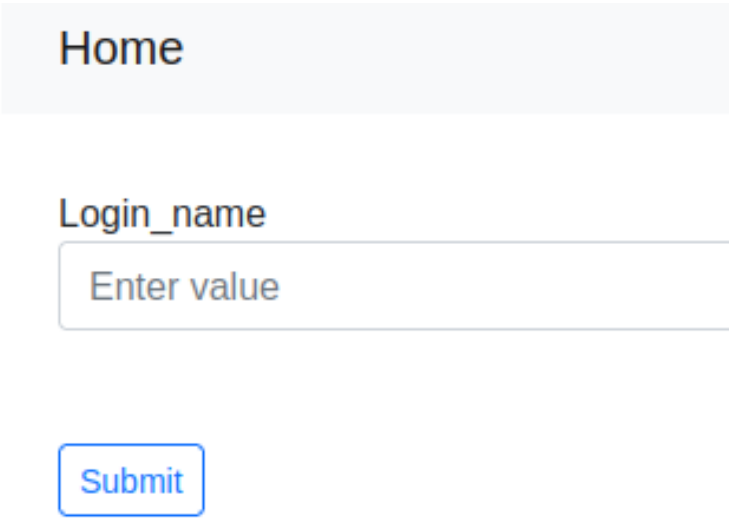
Adding new data to different tables looks easy and likewise. In every table that has foreign key field to another, adding a new row requires choosing foreign key that is related to that table.

Add operation constraints:

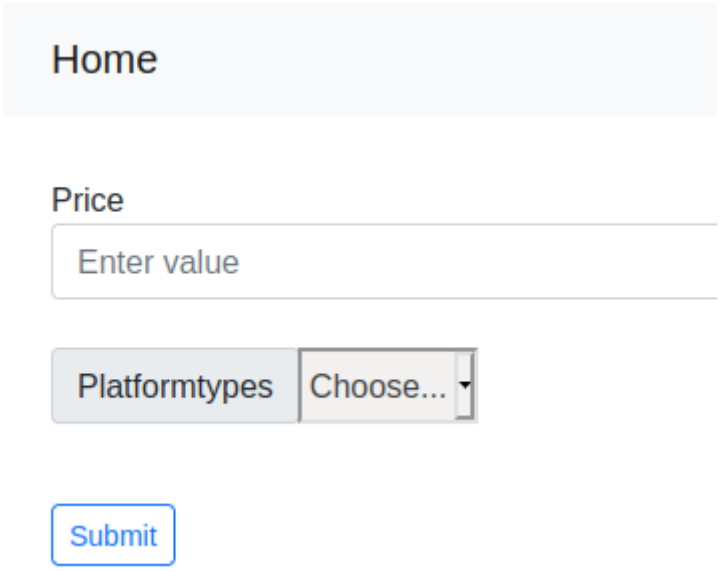
1. Duplicate data can’t be added in table which columns have unique constraints
2. Wrong data can’t be added. For example, invalid price (negative or invalid input) or invalid customer play time (negative or invalid input), blank data.
3. The new customer name in `Customers` table should have at least 4 symbols and do not fully consist of numbers.



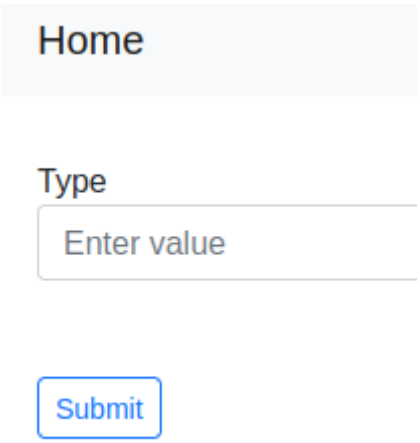
Pic. 3.1 Add new row in `Orders` table



Pic. 3.2 Add new row in `Customers` table



Pic. 3.3 Add new row in `Platforms` table



Pic. 3.4 Add new row in `Platformtypes` table

`Orders` table rows has aggregate field `price`, which calculates by joining tables on current FK and looks like:

*Orders.price = Orders.play_time * Platforms.price*

Edit data

Editing already existing data is simple and mutually exclusive between tables.
Edit operation constraints are completely the same with the add operations'.
Button `Edit` near every row in table performs edit action to that row.

If any prohibited action is performed, data will not be updated and a user will be redirected to the current table overview page.
The edit page for every table looks absolutely the same as adding a new row page.

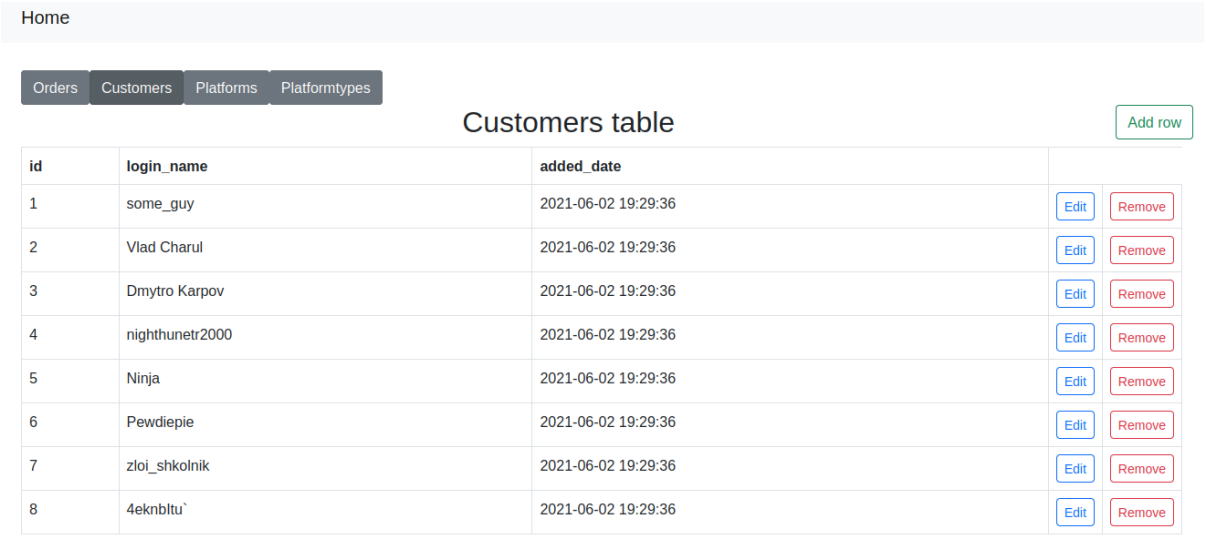
Delete data

Deleting the data is used for line deletion of a specific row in the table. User should choose the row to delete and, if the deletion is possible, it will proceed.

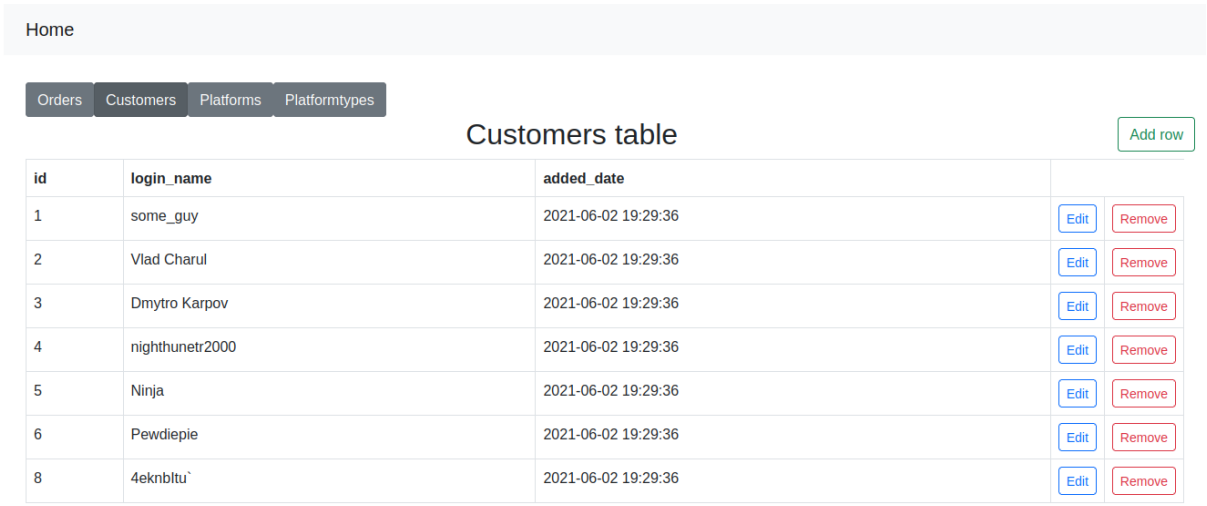
Delete operation constraint is only one. User can not delete a row which data are relates to data in another table.

If deletion is performed upon prohibited action, no changes will be made.

Button `Remove` near every row in the table performs delete action to that row.



4.1 `Customers` table before deletion row with id - 7



4.2 `Customers` table after deletion row with id - 7