Windows

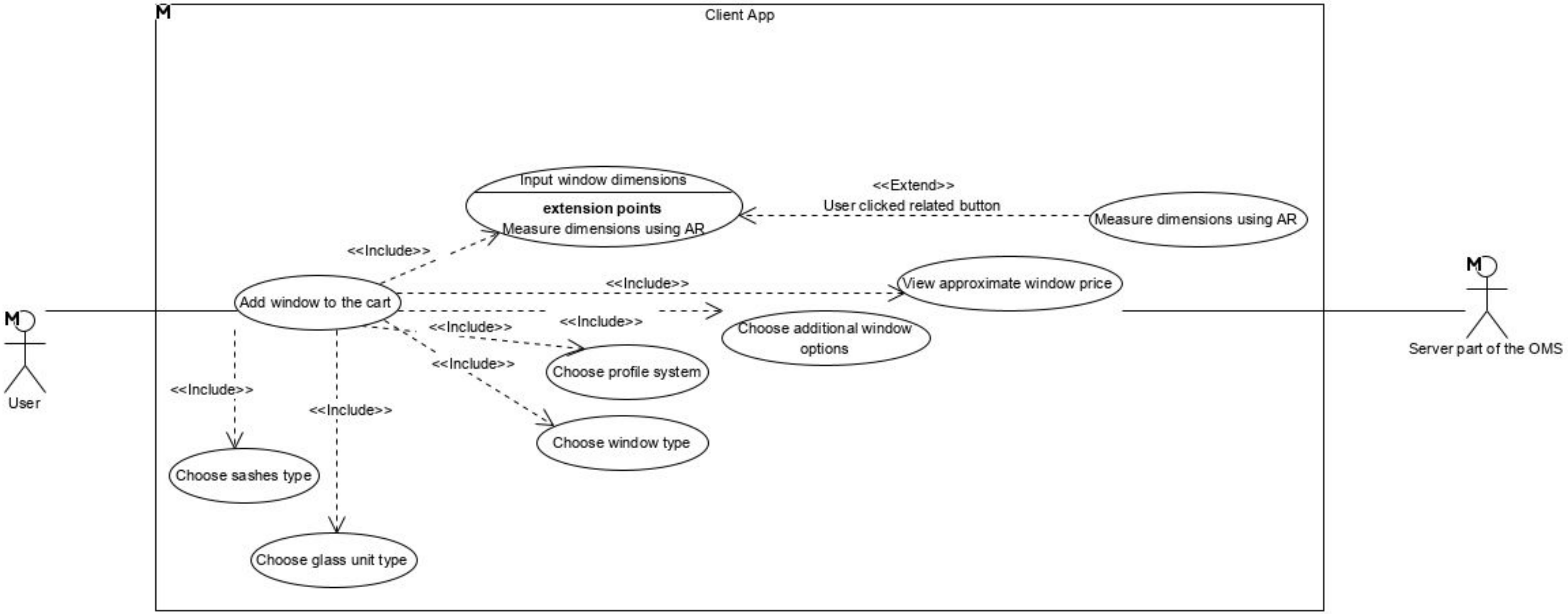
Data design

Product description

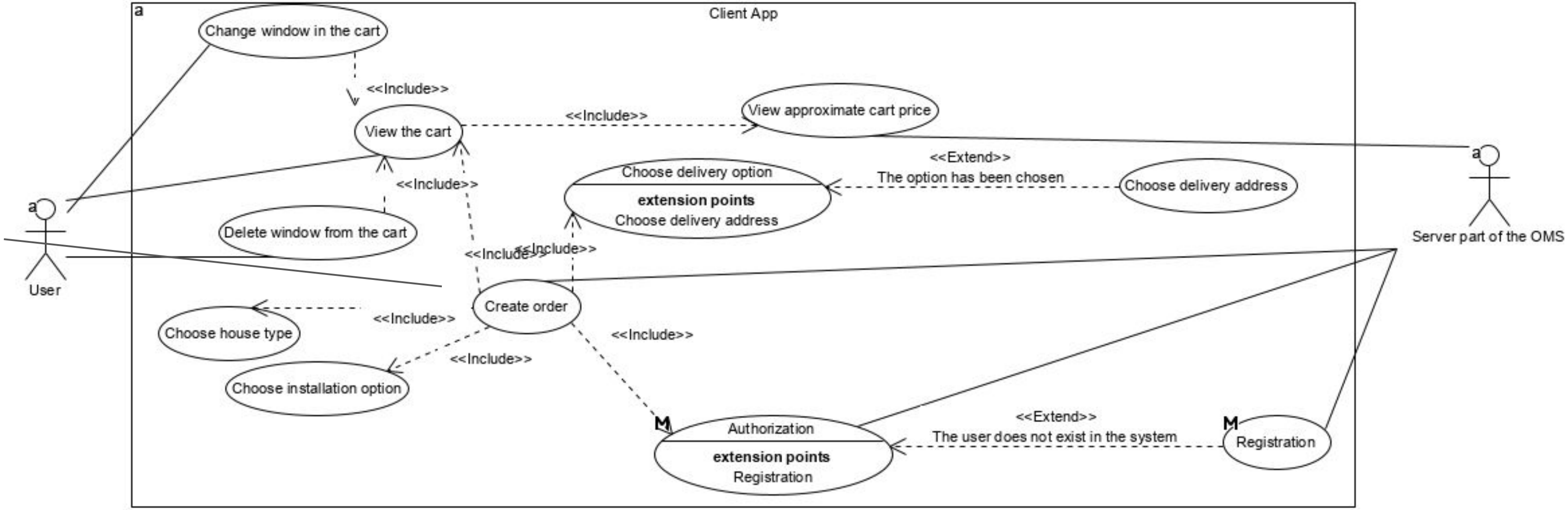
A platform designed to support the operation of a window manufacturing, sales and installation company, namely the process of interaction with the customer, designed to reduce the time and material losses of the company and to increase customer interest in it. The latter is intended to be achieved by including in the AR system technology for measuring the approximate size of windows, which will allow users to calculate the cost of a window before ordering and to select the best option for them.

Team: Alina Kolchanova, Asgar Zagitov, Semen Sokolov

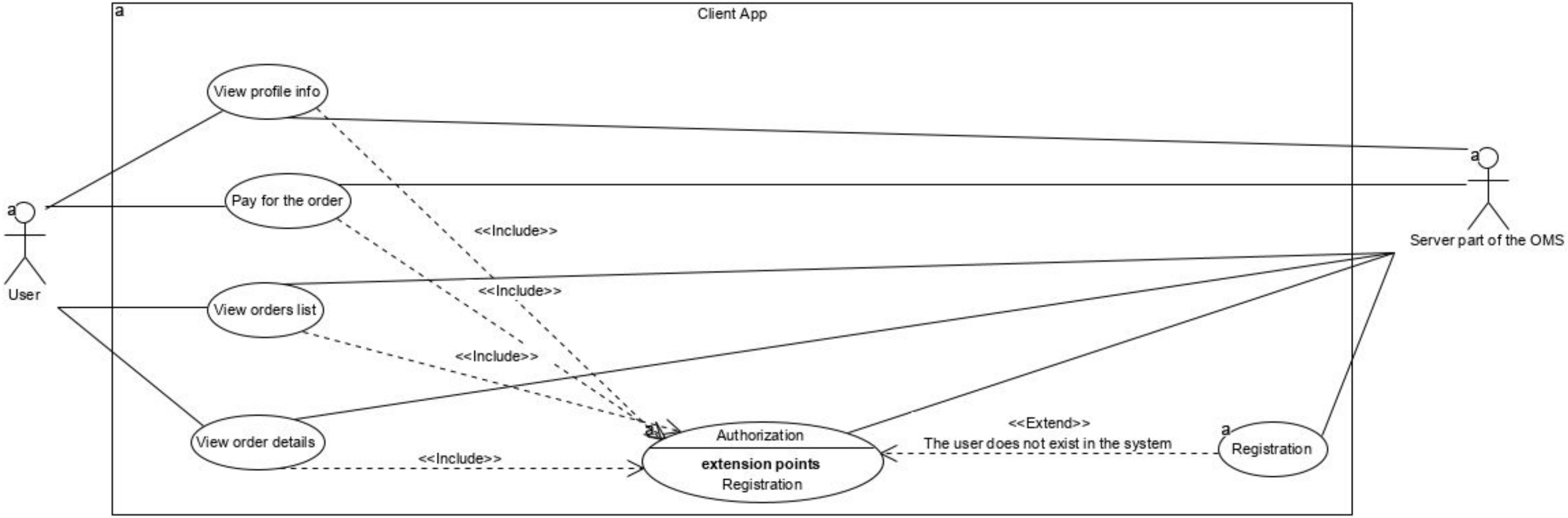
Use case diagram



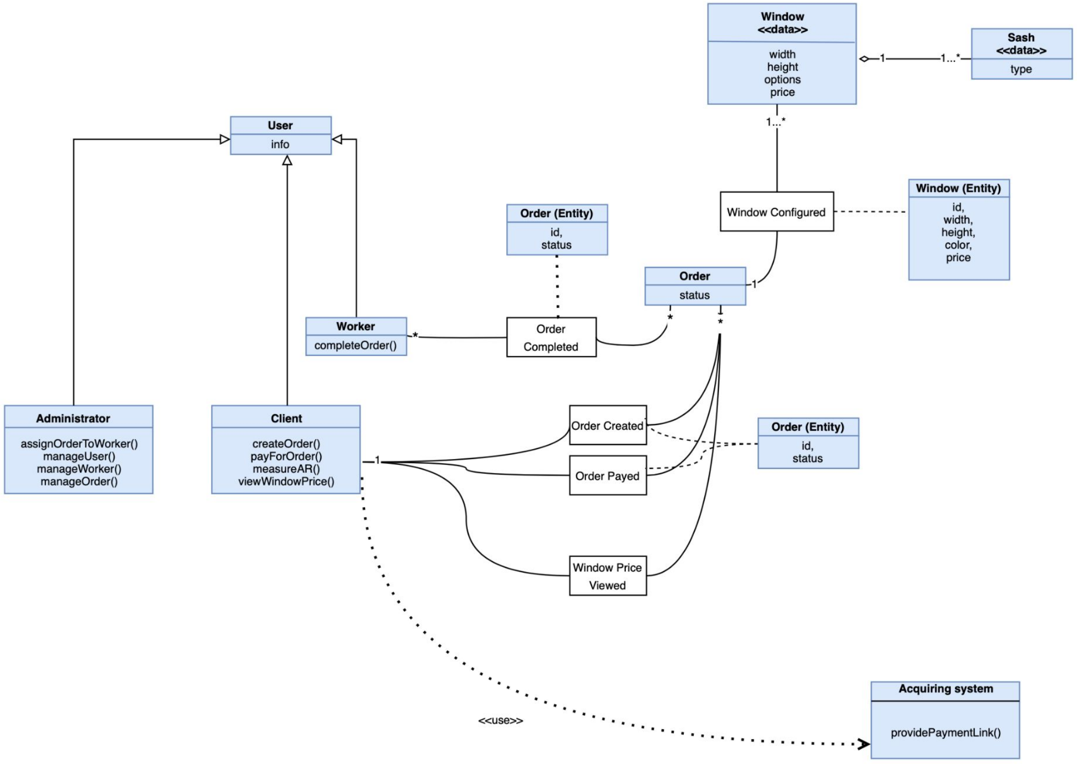
Use case diagram



Use case diagram



Detailed class diagram



Window
<<data>>
width
height
options
price

1
1...*

Sash
<<data>>
type

1...*

User
info

Window Configured

Window (Entity)
id,
width,
height,
color,
price

Order (Entity)
id,
status

Order
status

1

Worker
completeOrder()

*

Order
Completed

*

*

Administrator
assignOrderToWorker()
manageUser()
manageWorker()
manageOrder()

Client
createOrder()
payForOrder()
measureAR()
viewWindowPrice()

1

Order Created

Order Payed

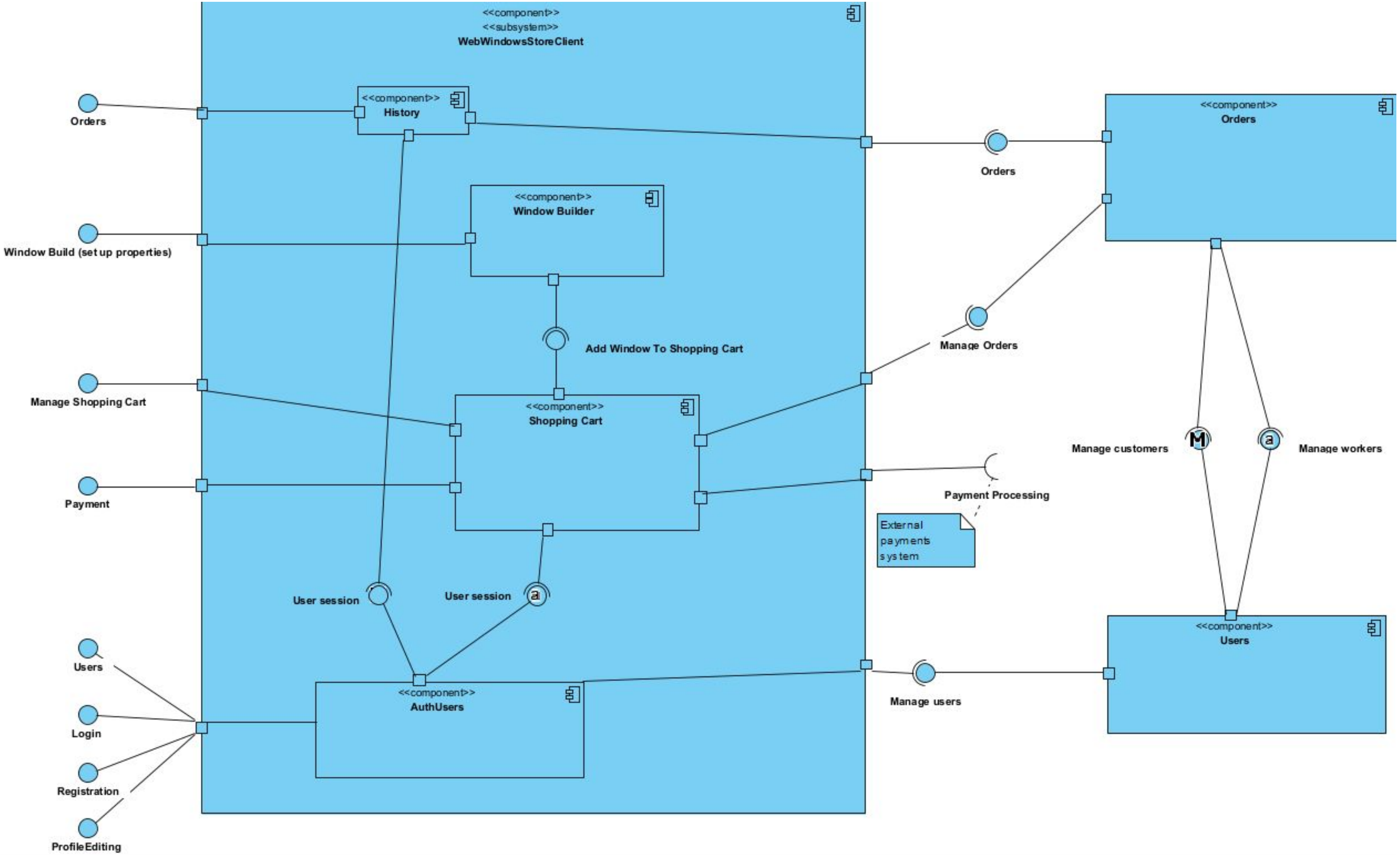
Order (Entity)
id,
status

Window Price
Viewed

<<use>>

Acquiring system
providePaymentLink()

Service diagram



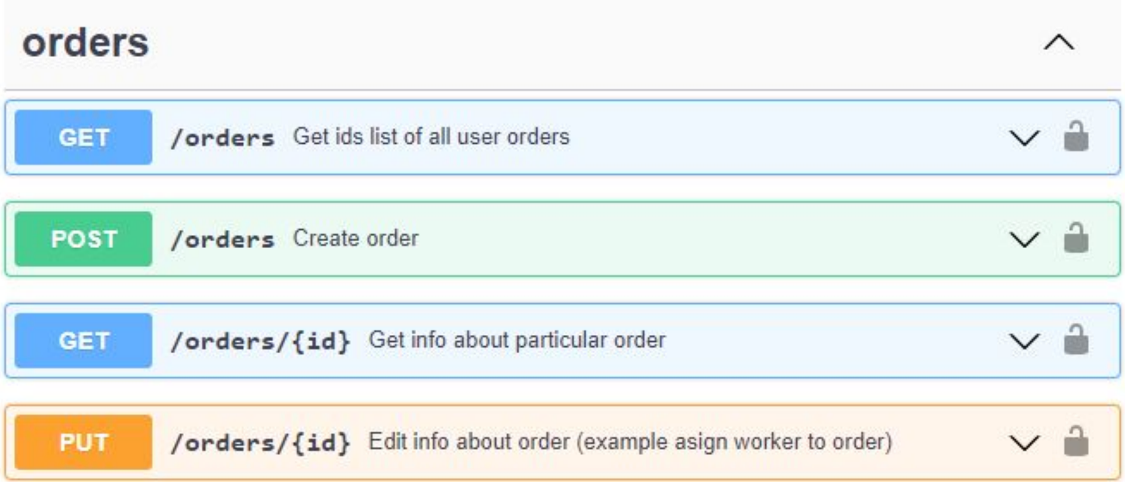
API usage Orders

User stories:

- when the user pays for the basket, the order information is recorded in the system
- when the user views the order history

Service: Orders service

Done by: Asgar Zagitov



orders

GET /orders Get ids list of all user orders

POST /orders Create order

GET /orders/{id} Get info about particular order

PUT /orders/{id} Edit info about order (example asign worker to order)

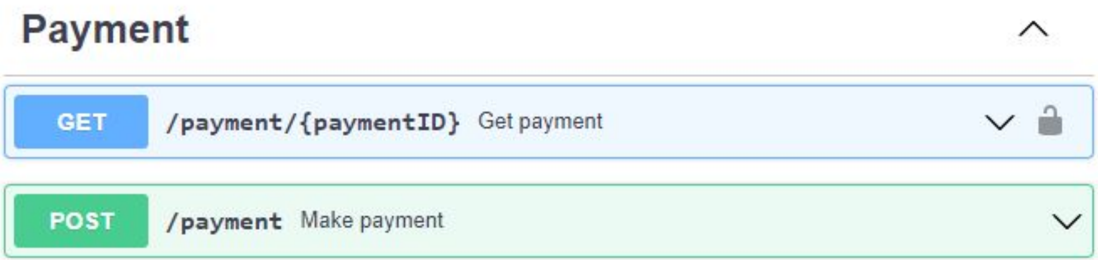
API usage Payment

User stories:

- User or worker checks user’s payment
- User makes payment after creating an order

Service: Payment service

Done by:
Alina Kolchanova



Payment

GET /payment/{paymentID} Get payment

POST /payment Make payment

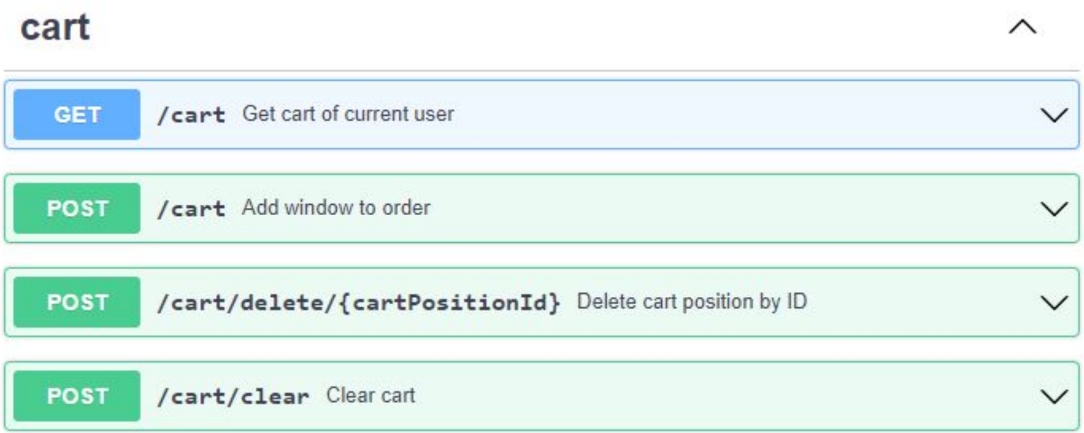
API usage Shopping Cart

User stories:

- Delete window from cart
- Create order form current cart

Service: Cart service

Done by:
Sokolov Semen



cart

GET `/cart` Get cart of current user

POST `/cart` Add window to order

POST `/cart/delete/{cartPositionId}` Delete cart position by ID

POST `/cart/clear` Clear cart

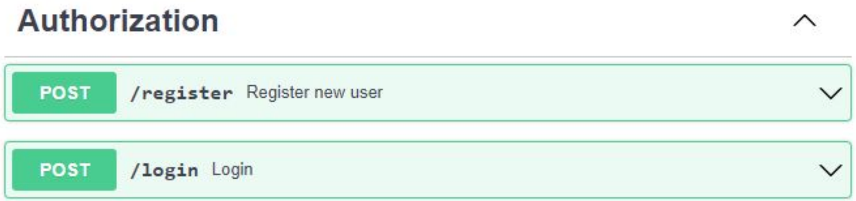
API usage Authorization

User stories:

- New user want to create account
- Old user want to do something as logged in user

Service: Auth service

Done by:
Asgar Zagitov



Authorization

POST /register Register new user

POST /login Login

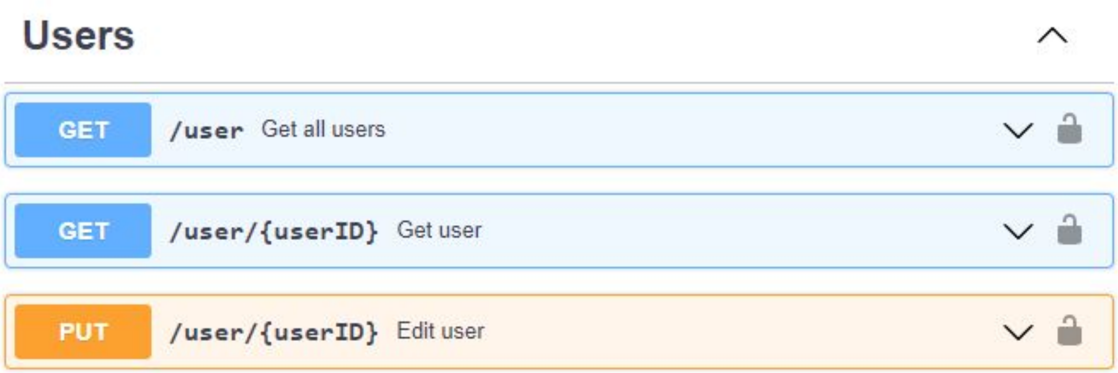
API usage User

User stories:

- User wants to check or change his info
- Owner wants to have some statistics

Service: User service

Done by:
Sokolov Semen



Users

GET /user Get all users

GET /user/{userID} Get user

PUT /user/{userID} Edit user

Choice of database technology

A relational database model was chosen for the project.

The reason is that the execution of ACID in the system is much more important than the storage of a huge amount of data and easy scalability. The data is also structured, with known properties and fields (windows, orders, etc.)

For development and operation, the PostgreSQL DBMS was selected. The reasons are as follows: a huge community, excellent documentation, support for all necessary SQL operations, relationality, ACID support. But at the same time, PostgreSQL can still be scaled in case of urgent need, which can help in the future development of the system.

Physical schema Order

```
CREATE TABLE windows.order (
order_id serial PRIMARY KEY,
status VARCHAR(15) NOT NULL
worker_id INT,
installation_option VARCHAR(50),
address VARCHAR(100),
house_type VARCHAR(20),
user_id INT,
window_id INT,
CONSTRAINT fk_user_id
        FOREIGN KEY (user_id)
                REFERENCES windows.user(user_id),
CONSTRAINT fk_window_id
        FOREIGN KEY (window_id)
                REFERENCES
                windows.window(window_id)
);
```

```
CREATE SEQUENCE windows.order_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;
```

Physical schema Payment

```
CREATE TABLE windows.payment (
payment_id serial PRIMARY KEY,
user_id INT,
CONSTRAINT fk_user_id
        FOREIGN KEY (user_id)
                REFERENCES windows.user(user_id)
);
```

```
CREATE SEQUENCE
windows.payment_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;
```

Physical schema Shopping Cart

```
CREATE TABLE windows.cart (
cart_id serial PRIMARY KEY,
user_id INT,
CONSTRAINT fk_user_id
        FOREIGN KEY (user_id)
                REFERENCES windows.user(user_id)
);
```

```
CREATE SEQUENCE
windows.cart_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;
```

Physical schema User

```
CREATE TABLE windows.user (
user_id serial PRIMARY KEY,
order_id INT
user_id INT,
payment_id INT
CONSTRAINT fk_order_id
        FOREIGN KEY (order_id)
                REFERENCES windows.order(order_id)
CONSTRAINT fk_payment_id
        FOREIGN KEY (payment_id)
                REFERENCES
                windows.payment(payment_id),
CONSTRAINT fk_cart_id
        FOREIGN KEY (cart_id)
                REFERENCES
                windows.payment(cart_id)
);
```

```
CREATE SEQUENCE
windows.user_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;
```

Physical schema Window

```
CREATE TABLE windows.window (
window_id serial PRIMARY KEY,
width INT,
height INT,
length INT,
sash_type VARCHAR(50),
CONSTRAINT fk_order_id
        FOREIGN KEY (order_id)
                REFERENCES windows.order(order_id)
);
```

```
CREATE SEQUENCE
windows.window_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;
```

Team work

Order	Alina Kolchanova
Payment	Alina Kolchanova
Shopping Cart	Asgar Zagitov
User	Asgar Zagitov
Window	Semen Sokolov