

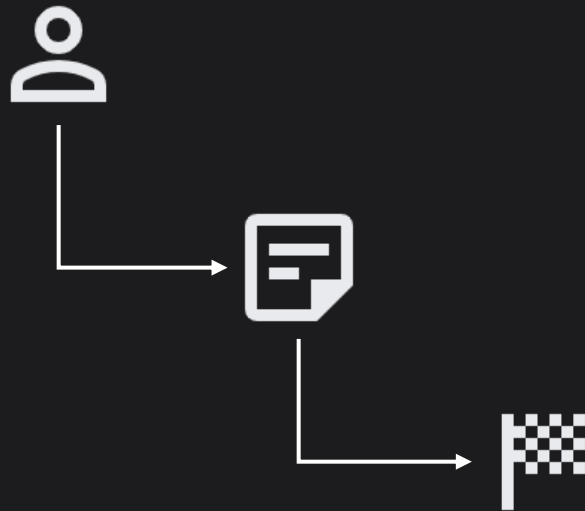
Water

A project by


Di Martino Ludovico, Galli Filippo, Leonardi
Alessandro, Rigobello Manuel, Scapinello Michele

WaCar


CAR EXPERIENCE MANAGEMENT SYSTEM




Cars list



Maserati GranTurismo Trofeo performance

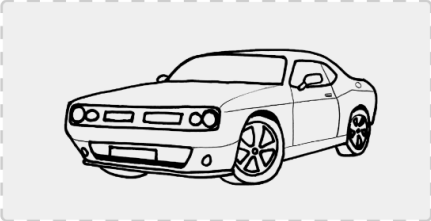


Lamborghini Huracan Tecnica supercar



Pagani Zonda R hypercar

Add new Car



Scegli file Nessun file selezionato

Select car type

Add new car type

☐ Available

Max Speed

Horsepower

0-100 (in seconds)

Submit

WACAR

WaCar offers you a platform to book an experience with your favourite car, on your favourite track! What are you waiting for?!!

OUR SITE

Logout

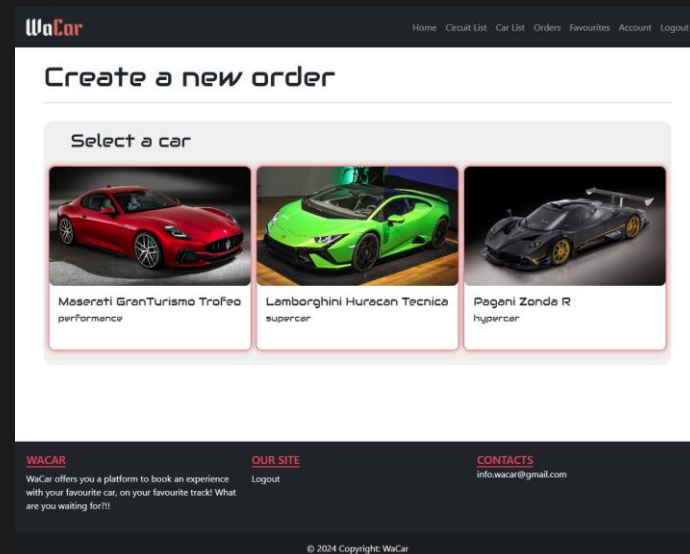
CONTACTS

info.wacar@gmail.com

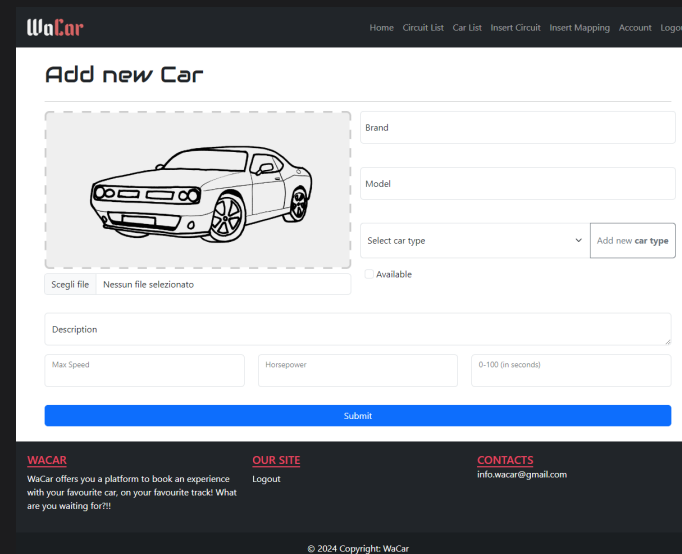
© 2024 Copyright: WaCar

Goals

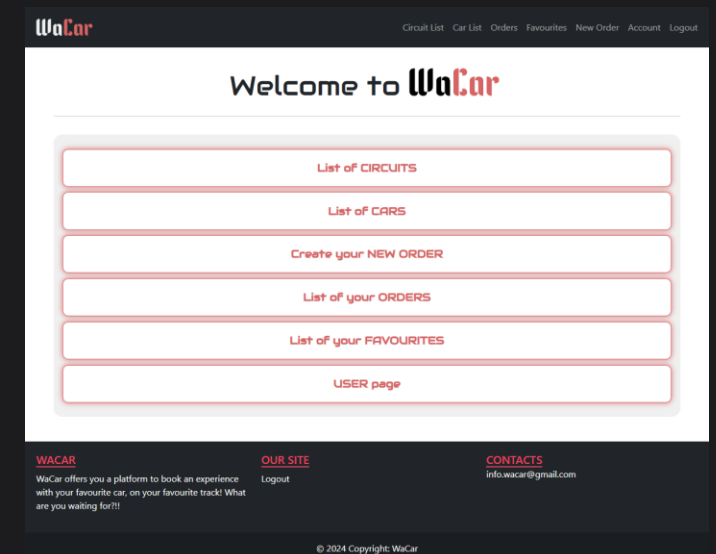
Easier way to
order



Easier management
of the products



User-Friendly &
responsive design



Types of User



Basic user

- Create and Manage her/his own orders



Administrator

- Administrator Features
- Cars and Circuits management

REST Resources

URI	DESCRIPTION
/rest/{car circuit}/	LIST all CARS/CIRCUITS
/rest/user/order/create/{carType}	GET Circuits based on {CarType}
/rest/user/order/create/complete	Create the Order
/rest/user/order/{orderId}	LIST ORDER based on the ID
/rest/user/favourite/{add delete}	CREATE/DELETE favourites ORDER
/rest/circuit/{circuitName}	GET circuit with name={circuitName}
/rest/car/{carBrand}/{carModal}	GET car with brand={carBrand} and modal={carModal}
/rest/user/favourite/search/{circuitName}/{carBrand}/{carModel}	GET the favorite with the selected values of the logged in user

REST Flow

REQUEST → RESTDispatcherServlet → Rest Resource → DAO

```
/**
 * Performs the serving logic for listing cars. Retrieves a list of cars from the database
 * using a Data Access Object (DAO) and sends the JSON representation of the list to the response output stream.
 * If successful, sets the HTTP status code to SC_OK (200). If an error occurs during the database access,
 * sets the HTTP status code to SC_INTERNAL_SERVER_ERROR (500) and sends an error message JSON response.
 *
 * @throws IOException If an error occurs while writing JSON data to the response output stream.
 */
+ Michele Scapinello +1
@Override
protected void doServe() throws IOException {
    Message m = null;
    List<Car> cars = null;
    try {

        // creates a new DAO for accessing the database and lists the cars
        cars = new ListCarDAO(con).access().getOutputParam();

        if (cars != null) {
            LOGGER.info("Car(s) successfully listed.");
            LOGGER.info(cars.getFirst().getModel());

            res.setStatus(HttpServletResponse.SC_OK);
            new ResourceList<>(cars).toJSON(res.getOutputStream());
        } else {
            LOGGER.error("Fatal error while listing car(s).");

            m = new Message(
                ErrorCode.UNEXPECTED_ERROR.getErrorMessage(),
                ErrorCode.UNEXPECTED_ERROR.getErrorCode(),
                errorDetails: null);

            res.setStatus(ErrorCode.UNEXPECTED_ERROR.getHTTPCode());
            m.toJSON(res.getOutputStream());
        }
    }
}
```

```
/**
 * Checks whether the request is for visualizing an {@link Order} and, in case, processes it.
 *
 * @param req the HTTP request.
 * @param res the HTTP response.
 * @return {@code true} if the request was for a {@code Product}; {@code false} otherwise.
 * @throws Exception if any error occurs.
 */
1 usage + manuel_rigobello +1
private boolean processOrder(final HttpServletRequest req, final HttpServletResponse res) throws Exception {
    final String method = req.getMethod();

    String path = req.getRequestURI();
    Message m = null;

    LOGGER.info("I am here");

    // the requested resource was not an order
    if (path.lastIndexOf(str: "user/order") <= 0) {
        LOGGER.info("Return false");
        return false;
    }

    // strip everything until after the /order
    path = path.substring(beginIndex: path.lastIndexOf(str: "order") + 5);
}
```

Generic Flow

Post/Get



```
protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
    LogContext.setIPAddress(req.getRemoteAddr());
    LogContext.setResource(req.getRequestURI());
    LogContext.setAction(Actions.GET_ALL_CARS);

    String op = req.getRequestURI();
    LOGGER.info("message: " + op, op);

    User user = (User) req.getSession().getAttribute(LoginFilter.ACCOUNT_ATTRIBUTE);
    if (user != null) {
        req.setAttribute("accountType", user.getType());
        LogContext.setUser(user.getEmail());
    } else {
        LogContext.setUser("NOT_LOGGED");
    }

    List<Car> cars = null;
    Message m = null;

    try {
        cars = new ListCarDAO(getConnection()).access().getOutputParam();

        m = new Message("Cars successfully retrieved");

        LOGGER.info("Cars successfully retrieved from database without parameters");
    } catch (SQLException ex) {
        m = new Message("message: " + "Cannot search for cars: unexpected error while accessing the database.", "errorCode: " + "E200", ex.getMessage());

        LOGGER.error("message: " + "Cannot search for cars: unexpected error while accessing the database.", ex);
    }
}
```

Servlet



```
public class ListCarDAO extends AbstractDAO<List<Car>> {
    1 usage
    private static final String STATEMENT = "SELECT * FROM assessment.car";

    /** Creates a new DAO object. ... */
    2 usages 1 Michele Scapinello
    public ListCarDAO(Connection con) { super(con); }

    /** Executes the SQL query to retrieve all cars from the database and populates a list of Car objects with the results. ... */
    1 Michele Scapinello +1
    @Override
    protected void doAccess() throws Exception {
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        final List<Car> cars = new ArrayList<Car>();

        try {
            pstmt = con.prepareStatement(STATEMENT);

            rs = pstmt.executeQuery();
        }
    }
}
```

DAO

JSP

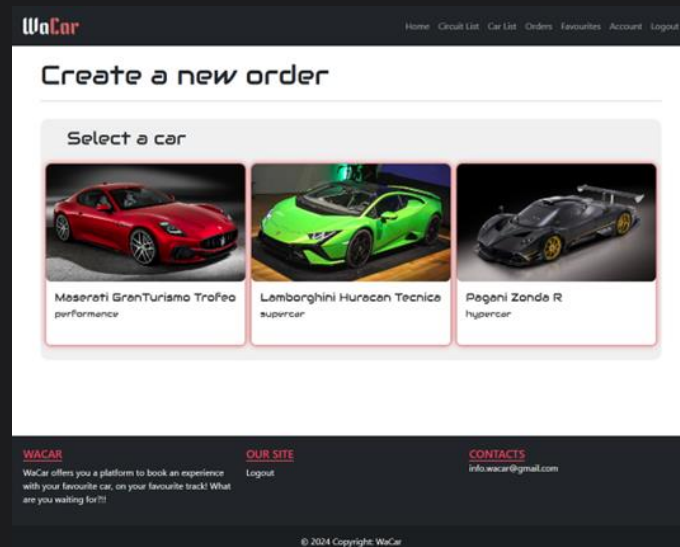


Return and
visualization

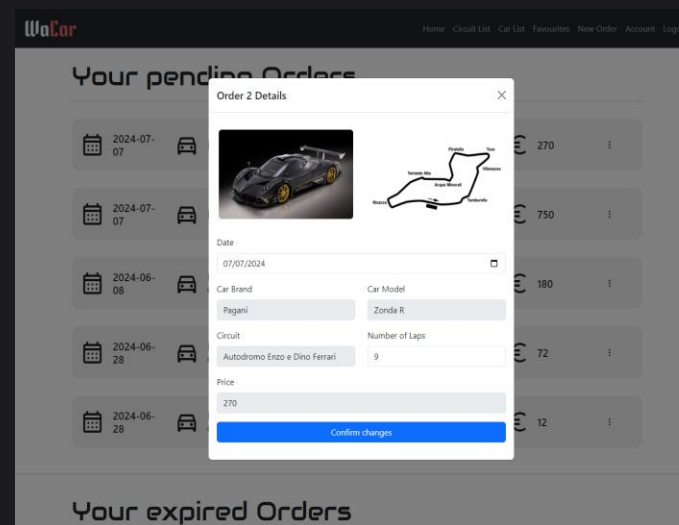
```
<div class="container">
    <h1>Cars list</h1>
    <div class="container">
        <div class="row">
            <div class="card">
                <img alt="Car image" data-bbox="250 700 350 750" data-toggle="modal" data-target="#carModal" />
                <div class="card-body">
                    <p>Car details</p>
                </div>
            </div>
        </div>
    </div>
    <div class="modal fade" id="carModal" tabindex="-1" role="dialog" aria-labelledby="carModalLabel" aria-hidden="true">
        <div class="modal-content">
            <div class="modal-body">
                <div class="card">
                    <div class="card-body">
                        <p>Car details</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

User Features

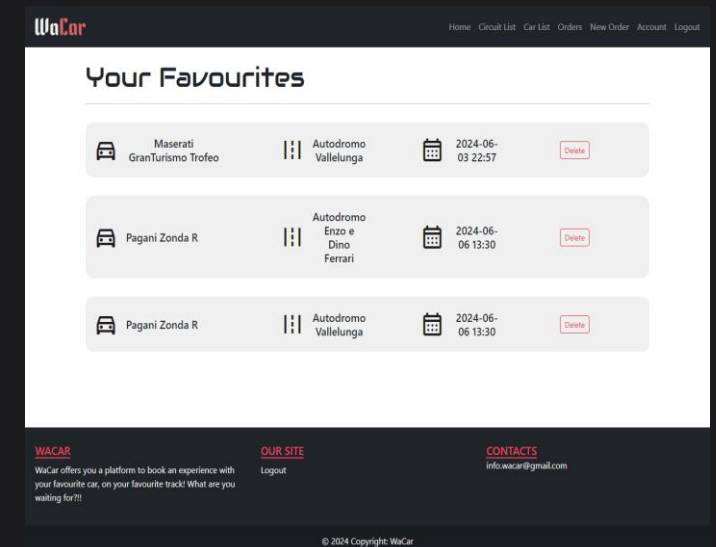
Create and manage Orders



1. Create

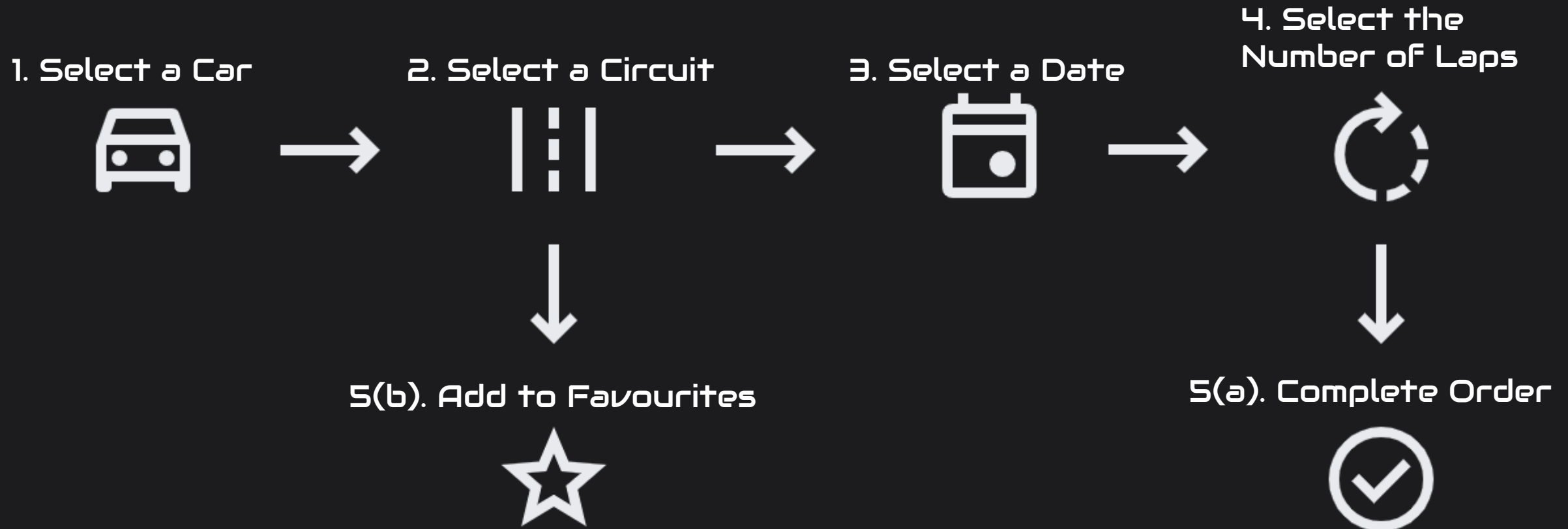


2. Modify/Delete



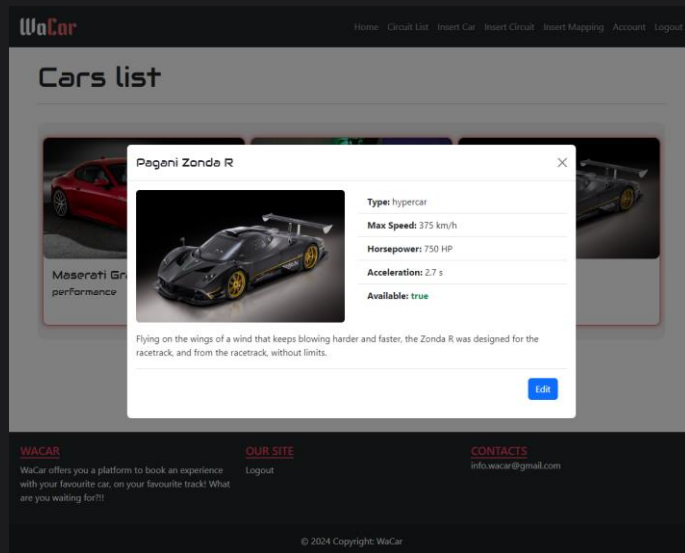
3. Favourites feature

Order procedure

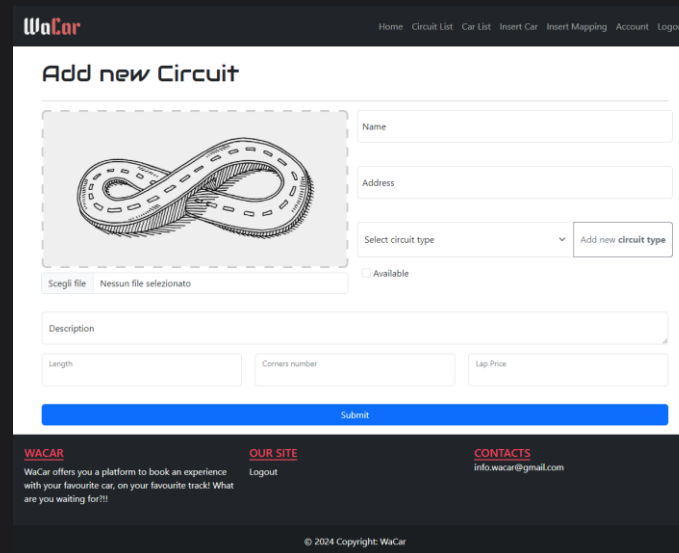


Admin Features

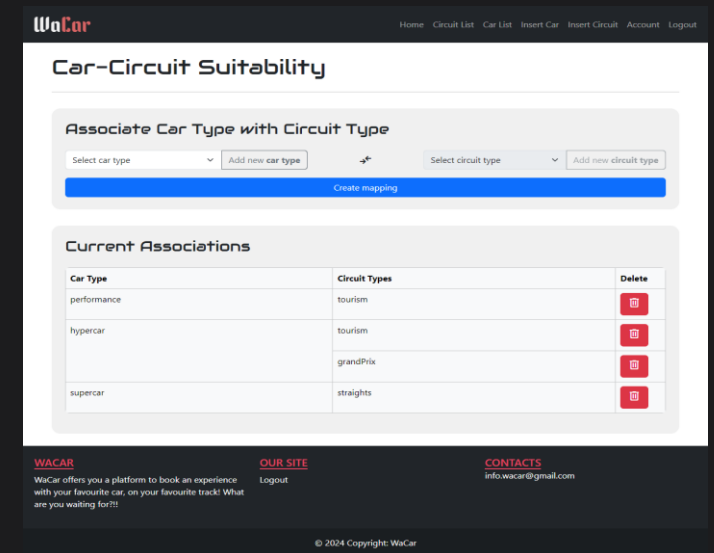
Cars/Circuits management



1. Edit

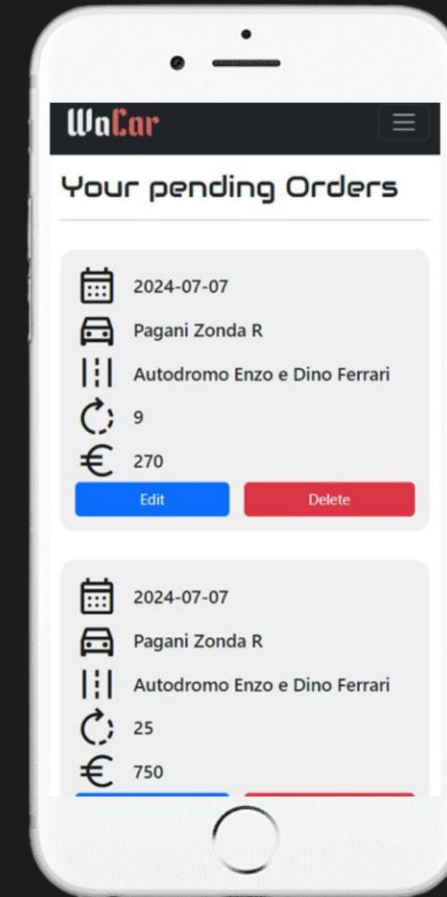
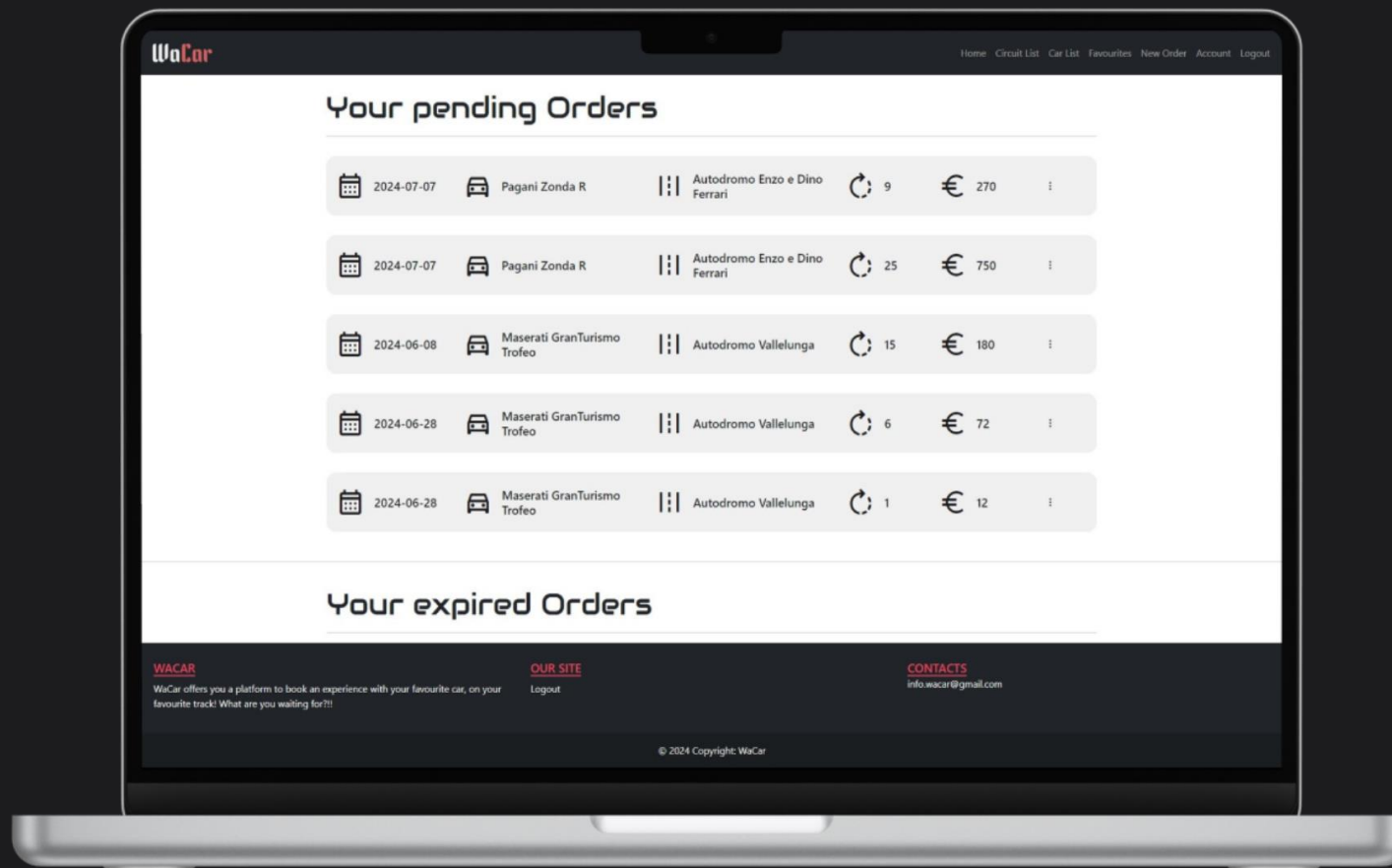


2. Insert



3. Map

Responsive Design



Conclusions

- WaCar satisfies the needs of customers and owners with its simple interface and usage
- The goal of the project has been achieved by providing a solution that facilitates the book of such experiences and the management of the offered products

Contributions

- **Di Martino Ludovico**: developed the frontend of the admin section and the order recap modal for the basic user. Created and added the favicon and the logo on all the pages of the website.
- **Galli Filippo**: developed the frontend for the login, the signup and the modify-order modal. Furthermore I have developed the UpdateOrder servlet and the navigation bar
- **Leonardi Alessandro**: developed the frontend of the list favourite page. Contributed to the development of the frontend of the list order page.
- **Rigobello Manuel**: developed the frontend of the list car, list circuit, their modals (with “edit” handling for the admin) and their respective REST. Developed the frontend of the create order page (no recap modal), the REST for getting the circuits based on the carType and the second modal with success/error message (error can arrive from PostgreSQL trigger). Developed the basic user Authentication header with BASIC authorization.
- **Scapinello Michele**: developed the frontend of the list order page and the delete order modal and servlet. Development of the footer and part of the navbar. Preparation of the slides.