**Прокат авто**

В системі існує перелік Автомобілів, для якого необхідно реалізувати:

- вибір по марці;

- вибір по класу якості;

- сортування за вартістю прокату;

- сортування за назвою.

Клієнт реєструється в системі, обирає автомобіль та робить замовлення на оренду. Незареєстрований клієнт не може зробити замовлення. В даних замовлення клієнт вказує паспортні дані, опцію 'з водієм'/'без водія', термін оренди. Система формує Рахунок, який клієнт оплачує.

Менеджер розглядає замовлення і може відхилити його, вказавши причину. Також менеджер реєструє повернення автомобіля, у випадку пошкодження автомобіля виставляє через систему рахунок на ремонт.

Адміністратор системи має права:

- додавання, видалення автомобілів, редагування інформації про автомобілі;

- блокування/розблокування користувачів;

- реєстрація менеджерів в системі.

**Вимоги до реалізації**

На основі сутностей предметної області створити класи, які їм відповідають.

Класи і методи повинні мати назви, що відображають їх функціональність, і повинні бути рознесені по пакетам.

Оформлення коду має відповідати Java Code Convention.

Інформацію щодо предметної області зберігати у реляційній базі даних (в якості СУБД рекомендується використовувати MySQL або PostgreSQL).

Для доступу до даних використовувати JDBC API із застосуванням готового або ж розробленого самостійно пулу з'єднань.

НЕ допускається використання ORM фреймворків.

Застосунок має підтримувати роботу з кирилицею (бути багатомовним), в тому числі при зберіганні інформації в базі даних:

повинна бути можливість перемикання мови інтерфейсу;

повинна бути підтримка введення, виведення і зберігання інформації (в базі даних), записаної на різних мовах;

в якості мов обрати мінімум дві: одна на основі кирилиці (українська або російська), інша на основі латиниці (англійська).

Архітектура застосунка повинна відповідати шаблону MVC.

НЕ допускається використання MVC-фреймворків.

При реалізації бізнес-логіки необхідно використовувати шаблони проектування: Команда, Стратегія, Фабрика, Будівельник, Сінглтон, Фронт-контролер, Спостерігач, Адаптер та ін.

Використання шаблонів повинно бути обґрунтованим.

Використовуючи сервлети і JSP, реалізувати функціональність, наведену в постановці завдання.

Використовувати Apache Tomcat у якості контейнера сервлетів.

На сторінках JSP застосовувати теги з бібліотеки JSTL та розроблені власні теги (мінімум: один тег custom tag library і один тег tag file).

Реалізувати захист від повторної відправки даних на сервер при оновленні сторінки (реалізувати PRG).

При розробці використовувати сесії, фільтри, слухачі.

У застосунку повинні бути реалізовані аутентифікація і авторизація, розмежування прав доступу користувачів системи до компонентів програми. Шифрування паролів заохочується.

Використовувати Впровадити у проект журнал подій із використанням бібліотеки log4j.

Код повинен містити коментарі документації (всі класи верхнього рівня, нетривіальні методи і конструктори).

Застосунок має бути покритим модульними тестами (мінімальний відсоток покриття 40%). Написання інтеграційних тестів заохочуються.

Реалізувати механізм пагінації сторінок з даними.

Всі поля введення повинні бути із валідацією даних.

Застосунок має коректно реагувати на помилки та виключні ситуації різного роду (кінцевий користувач не повинен бачити stack trace на стороні клієнта).

Самостійне розширення постановки задачі по функціональності заохочується (додавання капчі, формування звітів у різних форматах, тощо)!

Використання HTML, CSS, JS фреймворків для інтерфейсу користувача (Bootstrap, Materialize, ін.) заохочується!

За три дні до моменту старту захистів проектів (інтерв’ю) необхідно підготувати у вигляді окремого файлу схему бази даних, а також надати посилання на репозиторій із проектом.

**Етапи розробки**

Виконати аналіз завдання, вивчити прикладну область, обміркувати роботу системи загалом.

Описати функціонал системи з урахуванням передбачуваної послідовності реалізації:

Базовий функціонал, який реалізується в першу чергу.

Необхідний функціонал, який реалізується на наступному етапі.

Додатковий функціонал, який бажано реалізувати для підвищення зручності використання, рівня безпеки, продуктивності і т.д.

Розширений функцінал, реалізація якого може бути корисна.

Розписати кілька сценаріїв використання з урахуванням різних ролей користувача.

Розробити БД:

Виконати проектування (концептуальне, логічне, фізичне).

Заповнити даними для тестування.

Перевірити на відповідність функціоналу, вибраного для реалізації в п.2.

Розробити базовий призначений для користувача інтерфейс з урахуванням сценаріїв п.3.

Створити прототип

Реалізувати працюючу систему з базовим функціоналом.

Перевірити функціональність.

Усунути помилки і вибрати функціонал для поліпшення прототипу.

Послідовно поліпшувати прототип до досягнення запланованої функціональності.