EPAM Systems, RD Dep.

Практические задания для тренинга

Task. Web-project

REVISION HISTORY					
Ver.	**Description of Change**	**Author**	**Date**	**Approved**	
				Name	**Effective Date**
<3.0>	Вторая версия	Игорь Блинов	<10.XI.2016>		

FinalWebProject

Разработать веб-систему согласно требованиям.

Общие требования к проекту:

Приложение реализовать применяя технологии Servlet и JSP.

Архитектура приложения должна соответствовать шаблонам Layered architecture и Model-View-Controller. Controller может быть только двух видов: контроллер роли или контроллер приложения.

Информация о предметной области должна хранится в БД:

- данные в базе хранятся на кириллице, рекомендуется применять кодировку utf-8
- технология доступа к БД – JDBC (только JDBC)
- для работы с БД в приложении должен быть реализован потокобезопасный пул соединений,использовать слово synchronized запрещено.
- при проектировании БД рекомендуется не использовать более 6-8 таблиц
- доступ к данным в приложении осуществлять с использованием шаблона DAO.

Интерфейс приложения должен быть локализован; выбор из двух или более языков: (английский, белорусский, и т д.).

Приложение должно корректно обрабатывать возникающие исключительные ситуации, в том числе вести их логи. В качестве логгера использовать Log4J или SLF4J.

Классы и другие сущности приложения должны быть грамотно структурированы по пакетам и иметь отражающую их функциональность название.

При реализации бизнес-логики приложения следует при необходимости использовать шаблоны проектирования (например, шаблоны GoF: Factory Method, Command, Builder, Strategy, State, Observer, Singleton, Proxy etc).

Для хранения пользовательской информации между запросами использовать сессию.

Для перехвата и корректировки объектов запроса (request) и ответа (response) применить фильтры.

При реализации страниц JSP следует использовать теги библиотеки JSTL, использовать скриплеты запрещено. Обязательным требованием является реализация и использование пользовательских тегов. Просмотр “длинных списков” желательно организовывать в постраничном режиме.

Валидацию входных данных производить на клиенте и на сервере.

Документацию к проекту необходимо оформить согласно требованиям javadoc.

Оформление кода должно соответствовать Java Code Convention.

Разрешается использовать технологию Maven.

Приложение должно содержать 3-4 теста JUnit, Mockito или EasyMock.

Приложение должно быть размещено на Bitbucket.com

Общие требования к функциональности проекта:

1) Авторизация(sign in) и выход(sign out) в/из системы.
2) Регистрация пользователя или добавление артефакта предметной области системы.
3) Просмотр информации (например: просмотр всех лотов аукциона, кредитных карт, счетов и т.д.)
4) Удаление информации (например: отмена заказа, медицинского назначения, отказ от букинга номера и т.д.)
5) Добавление и модификация информации (например: создать и отредактировать курс, создать и отредактировать заказ и т.д.)

Предметные области для разработки проекта.

1. Система **Интернет-провайдер**. **Администратор** создает(корректирует) тарифы. **Клиент** подписывается (изменяет) тариф, просматривет состояние своего счета и траффика, вносит абонентскую плату. **Администратор** управляет клиентами: изменяет тарифный план по запросу пользователя, ставит бан неплательщикам.

2. Система **Аукцион**. **Администратор** выставляет(снимает, блокирует) на торги лоты, управляет **Клиентами**. Поддерживается несколько видов аукционов: интернет-аукцион, английский аукцион, блиц-аукцион. **Клиент** может участвовать в нескольких аукционах, оплачивать покупки, предлагать **Администратору** лоты для продажи.

3. Система **Сеть LikeIT**. **Пользователь** создает сообщения с просьбой о помощи в некотором вопросе, отвечает на сообщения других **Пользователей**, корректирует свои сообщения и персональную информацию. За ответ на поставленный вопрос, **Пользователь** ставит оценку по некоторой шкале. Оценки других **Пользователей** составляют рейтинг конкретного **Пользователя**. **Администратор** создает (изменяет, удаляет) темы сообщений, управляет **Пользователями**.

4. Система **Тотализатор. Клиент** делает **Ставки** разных видов (победа, ничья, поражения, точный результат и пр.) на **Соревнования**. **Букмекер** устанавливает уровень выигрыша. **Администратор** управляет **Пользователями**, создает (изменяет) **Соревнования**, а также фиксирует их результаты.

5. Система **BlackJack** или **“21”**. **Администратор** осуществляет управление **Игроками**. **Игра** может происходить как между **Игроком** и **ИИ**, так и между двуми **Игроками**. **Игрок** может пополнять свой **Счет. Игроки** могут обмениваться сообщениями. Победа\поражение изменяют рейтинг **Игрока**.

6. Система **КиноРейтинг**. **Администратор** создает (управляет) список фильмов, сериалов. **Пользователь** выставляет оценку (один раз) фильму и может оставить отзыв. Его статус автоматически повышается(понижается) если после определенного числа оценок других **Пользователей**, если его оценка близка(далека) от общего рейтинга. **Администратор** управляет пользователями: повышает, понижает статус, ставит баны.

7. Система **Социологический Опрос**. **Администратор** создает (управляет) **СоцОпрос** из нескольких **Вопросов** из определенного списка тем. **Посетитель** просматривает список доступных **Опросов,** отвечает на **Вопросы**, просматривает персональную статистику. **Администратор** просматривает результаты **Опросов**.

8. Система **Заказ АудиоТреков**. **Клиент** заказывает и оплачивает **АудиоТрек(и)**. Оставляет отзывы об **АудиоТреке. Администратор** добавляет новые **АудиоТреки**, корректирует информацию о существующих, управляет **Клиентами**, назначая им бонусы, скидки и пр.

9. Система **Хостел**. **Клиент** заполняет **Заявку** на бронирование или на полную оплату указывая необходимое число мест. **Администратор** подтверждает (отвергает) поступившую **Заявку**, делает скидки постоянным клиентам, ставит бан клиентам нарушившим правила системы.

10. Система **Online-Аптека**. **Клиент** выбирает необходимый препарат из списка доступных. Заполняет форму **Заказа**, указывая количество и дозировку. **Клиент** оплачивает **Заказ**. **Фармацевт** управляет списком препаратов. Часть препаратов требует электронного рецепта, которые может назначать **Клиенту** только **Врач**. **Клиент** может сделать запрос **Врачу** на продление рецепта.

11. Система **HR**. **Соискатель** вакансии заполняет регистрационную форму. **Сотрудник HR** фиксирует результат предварительного собеседования с **Соискателем** и назначает при необходимости техническое собеседование, по результатам которого и принимается решение о трудоустройстве **Соискателя**. **Сотрудник HR** размещает сообщения о вакансиях и управляет ими.

12. Система **EPAM-cafe**. Клиент делает **Заказ** на обед (выбирает из меню) и указывает время когда он хотел бы получить заказ. Система показывает цену заказа и предлагает оплатить с клиентского счета или наличными при получении заказа. Клиенту за предварительные заказы начисляются баллы лояльности. Если **Клиент** делает заказ, и не забирает его, то баллы лояльности снимаются вплоть до его блокировки**. Клиент** может оценивать каждый Заказ и оставлять отзывы. **Администратор** управляет меню, выставляет\убирает баны **Клиентам**.