Сессия 1

Восстановление базы данных из скрипта

Для восстановления таблиц в созданную базы данных воспользуйтесь предоставленным скриптом (mysql-database.sql). Вы должны доработать эту базу данных, обеспечив хранение всех данных, которые предоставил Вам заказчик. Для этого вам необходимо будет добавить необходимые таблицы, сущности, атрибуты и связи.

Импорт данных

Заказчик системы предоставил файлы с данными (с пометкой import в ресурсах) для переноса в новую систему. Подготовьте данные файлов для импорта и загрузите в разработанную базу данных.

Разработка десктопных приложений

Авторизация

Первый экран разрабатываемого приложения – авторизация пользователей. На нем должны быть поля для ввода логина и пароля, кнопка авторизации. В системе должны быть предусмотрены 3 роли: администратор, продавец и покупатель. Администратору в дальнейшем должен быть доступен функционал редактирования и добавления данных, а продавцу и обычному пользователю – только просмотр данных. Также в форме должна присутствовать кнопка для входа в неавторизованном режиме, в котором пользователю также будет доступен только просмотр данных.

Список продуктов

Необходимо отобразить продукты в виде списка, который должен соответствовать макету, предоставленному в ресурсах к заданию (product_list_layout.jpg). Записи для списка должны быть подгружены из базы данных.

У каждого продукта должны отображаться следующие данные: наименование, тип, логотип (картинкой), стоимость, дата сертификации и описание. При отсутствии изображения логотипа или возможности его загрузить необходимо выводить картинку-заглушку из ресурсов (picture.png).

Реализуйте сортировку списка продуктов по стоимости. Сортировка должна работать как по возрастанию, так и по убыванию. Для сортировки должен быть создан только один управляющий элемент, который будет менять направление сортировки при каждом его использовании. Сортировка должна быть реализована отдельно (не с помощью стандартных функций элементов управления).

Реализуйте фильтрацию списка продуктов по типу. Все уникальные типы должны быть выведены в выпадающий список для фильтрации. Первым элементом в выпадающем списке должен быть “Все типы”, при выборе которого настройки данного фильтра сбрасываются.

Все фильтрации должны работать в реальном времени (то есть без необходимости нажатия кнопки “найти”).

Реализуйте поисковую строку, через которую пользователь сможет искать продукты по названию. Поиск должен быть реализован в реальном времени без необходимости нажатия кнопки по типу «Найти».

Сортировки, фильтрации и поиск должны применяться к списку одновременно. Также необходимо реализовать кнопку, которая будет сбрасывать все фильтры, сортировки, а также очищать поле поиска. После нажатия этой кнопки данные в таблице должны отображаться в начальном, неотфильтрованном и неотсортированном виде.

В нижней части окна необходимо показывать количество выведенных записей и их общее количество в базе. Например, 155 из 237. В случае если данные в таблицу выводятся после фильтрации или поиска, количество выведенных данных необходимо обновить исходя из размера выборки.

Добавление/редактирование/удаление продуктов

Необходимо добавить возможность редактирования данных существующих продуктов, а также добавление новых в отдельных окнах. Переходы на данные окна должны быть реализованы из главной формы со списком: для редактирования - при двойном нажатии на конкретную запись, для добавления - при нажатии кнопки “Добавить новый продукт”. Не должно открываться несколько форм добавления или редактирования одновременно.

В форме добавления должны быть предусмотрены следующие поля: наименование, тип (выпадающий список), путь до изображения, стоимость, дата сертификации и описание (с возможностью многострочного ввода). Стоимость продукта может включать сотые части, а также не может быть отрицательной. Ввод даты сертификации должен быть реализован с помощью календаря либо группы из 3-х выпадающих списков, позволяющих выбрать год (от 1991 до 2021), месяц и день. Форма редактирования должна содержать все поля из формы добавления и дополнительно поле с уникальным идентификатором.

Для всех полей, которые пользователь заполняет без использования готовых средств (выпадающие списки, календари, спиннеры и тд) должны быть реализованы проверки на корректность введенных данных. Если хотя бы одно поле не заполнено или заполнено некорректными данными, то при попытке добавления/сохранения продукта должно выводиться соответствующее уведомление.

При добавлении нового продукта уникальный идентификатор должен автоматически генерироваться в то время, как поле для него должно отсутствовать. Когда пользователь редактирует продукт, то поле с уникальным идентификатором должно быть доступно только для чтения.

В окне редактирования должна присутствовать кнопка “Удалить”, которая удаляет продукт из базы данных. При ее нажатии у пользователя необходимо запросить подтверждение действия. Также при удалении должно соблюдаться следующее условие: если у продукта есть информация о производстве (записи в таблице «ProductMaterial»), то их необходимо удалить вместе с продуктом, а пользователю вывести уведомление о количестве таких удаленных записей.

При открытии формы для редактирования все поля выбранного объекта должны быть подгружены из базы данных в соответствующие поля ввода.

После успешного редактирования, добавления или удаления продукта данные о нем в окне списка продуктов должны быть обновлены.

Тестирование

Все как обычно, разработайте библиотеку классов, бла, бла, бла… Мне лень все это еще раз расписывать, перейдем сразу к делу. В примере ниже используются классы (LocalTime), с которыми мы не работали до этого, чуть позже я запишу подробный гайд с разбором.

Кто сможет сам разобраться, то почитать можно тут
https://javadevblog.com/polnoe-rukovodstvo-po-java-8-date-time-api-primery-localdate-instant-localdatetime-parse-i-format.html

Разработайте библиотеку (отдельным проектом), которая позволит вернуть список свободных временных интервалов (заданного размера) в графике сотрудника для формирования оптимального графика работы сотрудников.

Название .jar файла	UserWorkTimeLib
Название класса	Calculation
Название метода	getPeriods
Входящие параметры	список занятых промежутков времени (в двух массивах: startTimes - начало, durations - длительность) LocalTime[] startTimes int[] durations минимальное необходимое время для работы менеджера int consultationTime рабочий день сотрудника (начало и конец) LocalTime beginWorkingTime LocalTime endWorkingTime
Возвращаемые тип данных	список подходящих свободных временных промежутков (в массив строк форматаHH:mm-HH:mm) String[]

Пример

Входные данные	Выходные данные
startTime | duration 10:00 60 11:00 30 15:00 10 15:30 10 16:50 40 consultationTime 30 beginWorkingTime 08:00 endWorkingTime 18:00	08:00-08:30 08:30-09:00 09:00-09:30 09:30-10:00 11:30-12:00 12:00-12:30 12:30-13:00 13:00-13:30 13:30-14:00 14:00-14:30 14:30-15:00 15:40-16:10 16:10-16:40 17:30-18:00