# Задание на разработку тестового приложения

## Общие положения

## Схема базы данных

Заказчик

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип поля | Описание |
| ID | Guid | Первичный ключ определяющий запись в таблице. Не пустое |
| NAME | Строка | Наименование заказчика. Не пустое. |
| CODE | Строка | Код заказчика. Содержит данные формата «ХХХХ-ГГГГ» где Х – число, ГГГГ – год в котором зарегистрирован заказчик. Не пустое. |
| ADDRESS | Строка | Адрес заказчика |
| DISCOUNT | Число | % скидки для заказчика. 0 или null – означает что скидка не распространяется. |

Заказ

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип поля | Описание |
| ID | Guid | Первичный ключ определяющий запись в таблице. Не пустое |
| CUSTOMER\_ID | Guid | ИД заказчика. Не пустое |
| ORDER\_DATE | Дата | Дата когда сделан заказ. Не пустое |
| SHIPMENT\_DATE | Дата | Дата доставки |
| ORDER\_NUMBER | Число | Номер заказа |
| STATUS | Строка | Состояние заказа |

Элемент заказа

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип поля | Описание |
| ID | Guid | Первичный ключ определяющий запись в таблице. Не пустое |
| ORDER\_ID | Guid | ИД заказа. Не пустое |
| ITEM\_ID | Guid | ИД товара. Не пустое |
| ITEMS\_COUNT | Целое число | Количество заказанного товара. Не пустое. |
| ITEM\_PRICE | Число | Цена за единицу. Не пустое. |

Товар

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип поля | Описание |
| ID | Guid | Первичный ключ определяющий запись в таблице. Не пустое |
| CODE | Строка | Код товара, формат «XX-XXXX-YYXX» где Х – число , Y- заглавная буква английского алфавита. Не пустое. |
| NAME | Строка | Наименование товара |
| PRICE | Число | Цена за единицу |
| CATEGORY | Строка(30) | Категория товара. |

Модель базы данных должна быть выполнена с использованием Entity Framework 4.0.

## Организация внутри приложения

В приложении должны присутствовать следующие слои:

Слой содержащий модель данных для отображения пользователю. Эти классы так же используются для получения информации от пользователя.

Слой для работы с данными. Слой должен содержать набор классов и методов для доступа к данным и операциями над ними. Операции в этом слое имеют элементарный характер. Например: извлечение информации о заказчике по его коду, получение заказов конкретного заказчика, получение информации о составе заказа по его номеру, сохранение и обновление информации.

Слой бизнес логики. Слой должен заниматься обработкой поступивших от пользователя данных (в виде модели), предварительной их обработкой и преобразованием в объекты для сохранения в базу данных. Так же слой занимается извлечением данных из базы данных , используя слой работы с данными, и преобразованием их в модели представления для выдачи конечному пользователю.

Примерная структура:

Запрос (Модель представления)

Ответ (модель представления)

Модель представления

Слой бизнес логики

Объет БД

Слой работы с данными

SQL запрос

База данных

## Взаимодействие с пользователем

Взаимодействие с пользователем осуществляется через Web интерфейс построенный на базе фреймворка VueJS3. Сайт должен иметь стандартную asp.net авторизацию. Каждый пользователь должен иметь свой логин и пароль для доступа к сайту. Каждый пользователь должен быть привязан к заказчику (за исключением администратора сайта). Операции по изменению данных должны осуществляться в виде диалоговых окон.

Интерфейс приложения должен иметь нейтральную стилистику. Не допускается использование контрастных переходов и ярких тонов.

## Инфраструктура приложения

Обмен между Web браузером и приложением осуществляется через формат данных JSON т.е. со стороны пользователя производятся операции и данные посылаются на сервер в формате JSON, сервер принимает данные и преобразует их в формат модели представления, затем передает бизнес слою для последующей обработки. Бизнес слой обрабатывает данные (валилирует, сохраняет или применяет другие операции) и возвращает ответ. Ответ преобразуется в JSON передается пользователю сделавшему запрос. Обмен данными между пользователем и сервером, а так же выполнение пользователем каких-либо действий, осуществляется с помощью AJAX запросов из браузера – серверу.

Примерный алгоритм работы. (Администратор входит в систему и добавляет новый вид товара)

1. Авторизация пользователя (пользователь вводит логин и пароль)
2. Система авторизирует пользователя, и если он является Администратором отображает стартовую страницу администратора.
3. Администратор выбирает пункт меню «Редактирваине каталога»
4. Отображается список товаров
5. Пользователь выбирает нужный товар и нажимает кнопку редактировать.
   1. Отправляется запрос к серверу на получение данных, AJAX запрос.
   2. Сервер возвращает данные для редактирования, формат JSON.
   3. Браузер отображает для редактиварония в модальном окне
6. Пользователь редактирует данные и нажимает «ОК»
   1. Данные собираются в форамте JSON и отправлябтся на сервр.
   2. Сервер проверяет и сохраняет данные. В случае успеха говорит что все успешно в противном случае отображает сообщение об ошибке.

## Функционал системы

### Роли

В системе должны 2 роли:

1. Роль заказчика
2. Роль менеджера

### Роль менеджера

1. Добавление, редактирование и удаление товаров
2. Добавление, редактирование и удаление пользователей
3. Подтверждение заказа. Устанавливается дата доставки и статус в Выполняется.
4. Закрытие заказа. Устанавливает статус Выполнен.

### Роль Заказчика

1. Просмотр каталога товаров.
2. Набирание товаров в корзину
3. Подача заказа товаров. Устанавливается дата создания и статус в Новый.
4. Просмотр статуса заказов. Фильтр по статусу.
5. Удаление заказа. Возможно до статуса Выполняется.

### Состояния заказа

1. Новый.
2. Выполняется.
3. Выполнен.

## Технологии и шаблоны проектирования

При разработке приложения желательно использовать следующие шаблоны проектирования:

1. Repository
2. Unit Of Work
3. MVC

Фреймфорки:

1. .Net Core 6-8
2. Entity Framework Core, Code First
3. VueJS 3