|  |  |
| --- | --- |
| «УТВЕРЖДАЮ»  От «Заказчика»  « » 2022 г.  М.П. | «УТВЕРЖДАЮ»  От «Исполнителя»  « » 2022 г.  М.П. |

**Автоматизированная информационная система «Склад»**

**Описание программы**

Казань 2022

# 1 ОПИСАНИЕ ПРОГРАММЫ

**1.1 ОБЩИЕ СВЕДЕНИЯ**

Наименование программы — «WarehouseSystem».

Для функционирования программы необходимы системные программные средства, предоставляемые операционной системой Windows, а также установленная СУБД Microsoft SQL Server 2019.

Программа написана на языке программирования C# с использованием Entity Framework Core (версия 6.0.1).

**1.2 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ**

**1.2.1 Классы решаемых задач**

Классы задач, которые помогает решать данная автоматизированная информационная система:

* Визуализация информации;
* Аналитика данных;
* Извлечение знаний;
* Интерпретация данных.

**1.2.2 Назначение программы**

Система предназначена для автоматизации деятельности работника склада и должна реализовывать следующий функционал:

* авторизация пользователя;
* обработку, хранение информации о товарах и поставщиках;
* фиксацию получения и выдачи экземпляров товара;
* ввод записей о поставщиках;
* выдачу в понятной форме результатов поиска;
* обеспечение защиты информации от несанкционированного доступа;

**1.2.3 Сведения о функциональных ограничениях на применение**

Программа для полноценной работы требует не менее 1 ГБ свободной памяти, не учитывая объёма памяти необходимого для размещения и хранения базы данных.

Помимо этого необходимо наличие всех ресурсов, поставляемых вместе с программой и предустановленная на компьютере СУБД Microsoft SQL Server с подключением к локальному серверу. Перед началом работы на этом сервере требуется создать соответствующую базу данных, с которой и будет работать программа.

**1.3 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ**

### **1.3.1 Алгоритм программы**

**1.3.1.1 Алгоритм авторизации**

1) Ввод пользователем авторизационных данных

2) Нажатие кнопки «Войти»

2.1) Запрос авторизационных данных из хранилища

2.2) Проверка на соответствие данных, введенных пользователем, данным, полученным из хранилища

2.3) Если данные совпали, переход к основному окну

2.4) Иначе, повторный ввод авторизационных данных (перейти к пункту 1)

1.3.1.2 Алгоритм добавления (получения) товара

1) Ввод данных о поставляемом товаре (наименование, город производства, дата производства и т.д.)

2) Нажатие кнопки «Получить товар»

2.1) Если поля не заполнены, то перейти к пункту 1

2.2) Иначе

2.2.1) Обработка данных

2.2.2) Отправка обработанных данных в хранилище

1.3.1.3 Алгоритм выдачи товара

1) Ввод артикула товара и количества выдаваемых экземпляров

2) Нажатие кнопки «Выдать»

2.1) Если поля не заполнены, перейти к пункту 1

2.2) Иначе

2.2.1) Обработка введенных данных

2.2.1.1) Если количество введенного товара больше, чем количество товара в хранилище, то выдача невозможна (перейти к пункту 1)

2.2.1.2) Если указанное количество совпадает с количеством экземпляров товара на складе, то удалить из базы данных заданное количество экземпляров и удалить товар

2.2.1.3) Иначе, выдать товар и удалить заданное количество экземпляров из хранилища.

1.3.1.4 Алгоритм добавления поставщика

1) Ввод имени, номера телефона и адреса поставщика

2) Нажатие кнопки «Добавить»

2.1) Если поля не заполнены, перейти к пункту 1

2.2) Иначе

2.2.1) Обработка данных

2.2.2) Отправка данных в хранилище

1.3.1.5 Алгоритм вывода поставщиков

1) Нажатие на кнопку «Вывести поставщиков»

2) Получение данных из хранилища

3) Вывод данных в пользовательский интерфейс

1.3.1.5 Алгоритм поиска товара

1) Ввод артикула либо наименования товара

2) Нажатие кнопки «Поиск»

3) Обработка ввода

3.1) Если поле пустое, вывод всех товаров

3.2) Если в поле введен артикул товар, вывод товаров с таким артикулом

3.3) Если в поле введено наименование товара, вывод товаров с таким наименованием

1.3.1.6 Алгоритм выхода

1) Нажатие кнопки «Выйти»

2) Посрать

### **1.3.2 Используемые методы**

Создание программы постоянно сопровождалось использованием следующих методик и парадигм программирования и проектирования:

* Объектно-ориентированное программирование;
* Объектно-ориентированная технология доступа к данным Entity Framework Core;
* Трехуровневая архитектура приложения.

**1.3.3 Структура программы с описанием функций составных частей и связи между ними**

Наша программа построена по принципу трехуровневой архитектуры приложения. Соответственно существуют следующие слои: Presentation (уровень представления), Core (уровень бизнес-логики), DAL (уровень доступа к данным).

Уровень представления данных – слой Presentation.

Назначение состоит в обеспечении наиболее эффективного обмена данными между пользователями и системой. Уровень представления реализует графический пользовательский интерфейс с простейшей бизнес-логикой (логический контроль вводимых данных и передача этих данных в слой Core (уровень бизнес-логики)).

Описание классов:

AuthorizationWindow – класс, отвечающий за ввод пользователем авторизационных данных и передаче данных в слой Core, для проверки на соответствие авторизационным данным, хранимым в хранилище.

MainWindow – класс, отвечающий за отображение основной формы приложения в которой происходит отображение основных данных по товарам, поиск товаров, а также переход к остальным формам приложения.

ProductDetailWindow – класс, отвечающий за отображение формы, в которой выводится детальная информация по определенному товару, получаемая из слоя Core.

ProductReceiveWindow – класс, отвечающий за отображение формы, в которую пользователь вводит данные поставляемого товара, а после, эти данные передаются в слой Core.

ProductIssueWindow – класс, отвечающий за отображение формы, в которой пользователь вводит данные товара, который надо выдать. После ввода, эти данные передаются в слой Core.

SupplierAddWindow – класс, отвечающий за отображение формы, в которую пользователь вводит данные нового поставщика, которого надо добавить. После ввода эти данные передаются в слой Core.

SupplierDetailWindow – класс, отвечающий за отображение формы, в которой происходит вывод детальной информации о всех поставщиках.

Program – класс, в котором происходит настройка приложения и всех зависимостей.

Уровень бизнес-логики – слой Core.

ProductBO – класс, описывающий продукт, как объект бизнес-логики.

ProductDetailsBO – класс, описывающий детальную информацию по продукту, как объект бизнес-логики.

ProductExemplarBO – класс, описывающий экземпляр продукта, как объект бизнес-логики.

SupplierBO – класс, описывающий поставщика, как объект бизнес-логики.

IProductService – интерфейс, который определяет некоторый функционал (набор методов) для реализации бизнес-логики приложения.

ProductService – класс, реализующий интерфейс IProductDervice, реализует всем методы, описанные в нем для реализации бизнес-логики приложении.

Уровень доступа к данным – слой DAL.

IProductRepository – интерфейс, который определяет некоторый функционал (набор методов) для реализации доступа к данным в хранилище.

ProductRepository – класс, реализующий интерфейс IProductRepository и все методы, описанные в нем для реализации доступа к данным.

Product – класс, описывающий таблицу Product в базе данных. Требуется, для корректного получения данных из хранилища.

ProductExemplar – класс, описывающий таблицу ProductExemplar в базе данных. Требуется, для корректного получения данных из хранилища.

Supplier – класс, описывающий таблицу Supplier в базе данных. Требуется, для корректного получения данных из хранилища.

Supply – класс, описывающий таблицу Supply в базе данных. Требуется, для корректного получения данных из хранилища.

WarehouseContext – промежуточный класс между базой данных и программным кодом, отвечает за связь с хранилищем.

Уровень представления (слой Presentation) связан с уровнем бизнес-логики (слой Core) посредством Dependency Injection (внедрения зависимости). Однако, слой Presentation ничего не знает об уровне доступа к данным (слой DAL). Слой Core связан со слоем DAL посредством внедрения зависимости. Слой DAL ничего не знает о слое Presentation. Уровень бизнес-логики получает данные из уровня представления данных, обрабатывает их посредством неких бизнес-правил и передает в уровень доступа к данным.

### **1.3.4 Связи программы с другими программами**

Для работы автоматизированной информационной системы необходимо прямое подключение к локальному серверу Microsoft SQL Server. Это соединение является единственным для системы. Программное подключение производится по «localhost» для этого сервера.

## **1.4 ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА**

Для функционирования программы необходим персональный компьютер (ПЭВМ), с минимальными техническими характеристиками:

* процессор с тактовой частотой, ГГц – 2, не менее;
* оперативную память объёмом, Гб-4, не менее;
* жёсткий диск со свободной памятью объёмом Гб-2, не менее (без учёта объёма памяти, необходимой для хранения базы данных).

Программа предназначена для ОС Windows 10 либо ОС Windows 11.

Помимо этого, для работы с автоматизированной информационной системой является обязательным наличие «мыши» и клавиатуры.

**1.5 ВЫЗОВ И ЗАГРУЗКА**

Программа «WarehouseSystem» поставляется в архиве (рис 1).



Рис.1

В архиве будет находится папка «WarehouseSystem». Требуется извлечь её из архива (рис. 2)

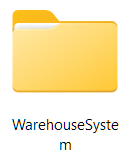


Рис.2

Далее открываем данную папку с помощью проводника. В папке находятся все необходимые для работы программы файлы (рис. 3)

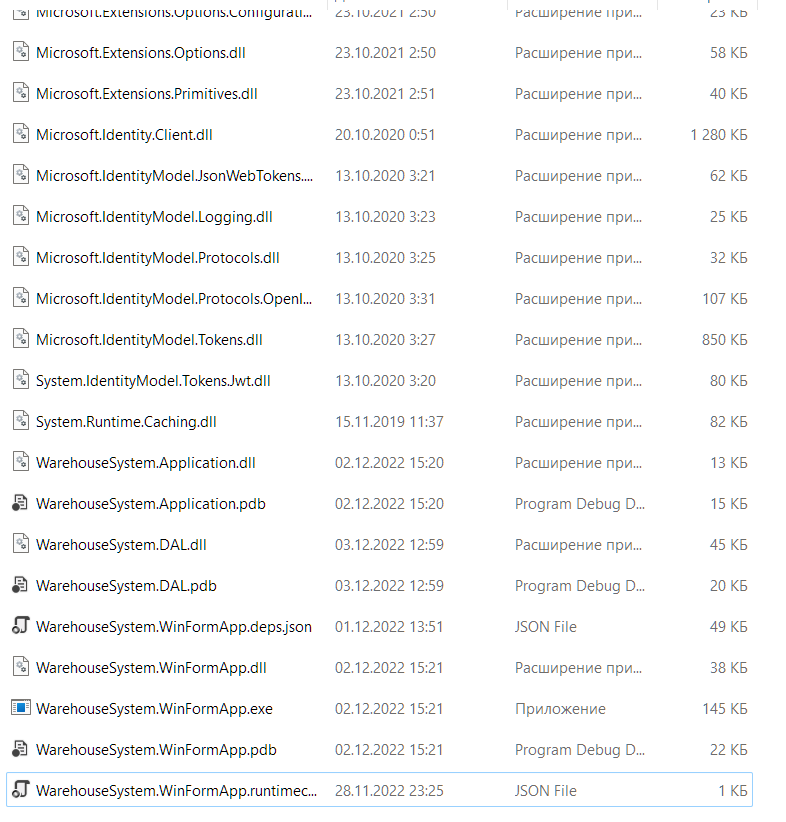


Рис. 3

Для запуска программы нужно запустить файл WarehouseSystem.WinFormApp.exe (рис. 4)

Точка входа в программу – фукция Main(). Данная функция реализована в файле Program.cs.

В данной функции мы подключаемся к локальному серверу базы данных Microsoft SQL Server при помощи метода CreateHostBuild().Build(), реализованного в файле Program.cs, а также запускаем и отображаем форму авторизации пользователя. После этого пользователь начинает работу в системе. Других точек входа в программу не предусмотрено.

## **1.6 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ**

Входные данные программы – данные, полученные из базы данных, а также данные, полученные через графический интерфейс приложения. Это данные: о товарах, о поставщиках – которые нужно внести в базу данных, различные сигналы передаваемые системе через пользовательский интерфейс, такие как нажатия кнопок, например.

Выходные данные программы – данные, передаваемые в базу данных программой, а также данные, которые выводятся пользователю через графический интерфейс приложения. Выходными данными являются данные: о товарах, о поставщиках. Также, это могу быть некие запросы к базе данных на вывод в неё новых данных, а также изменение уже имеющихся в ней данных.

Наиболее наглядно потоки входных и выходных данных отображает диаграмма потоков данных DFD. В наиболее общем виде эти потоки представлены в контекстной диаграмме системы с процессом ОБСЛУЖИТЬ, идентифицирующая внешние сущности СОТРУДНИК СКЛАДА и БАЗА ДАННЫХ СКЛАДА (рис. 5).

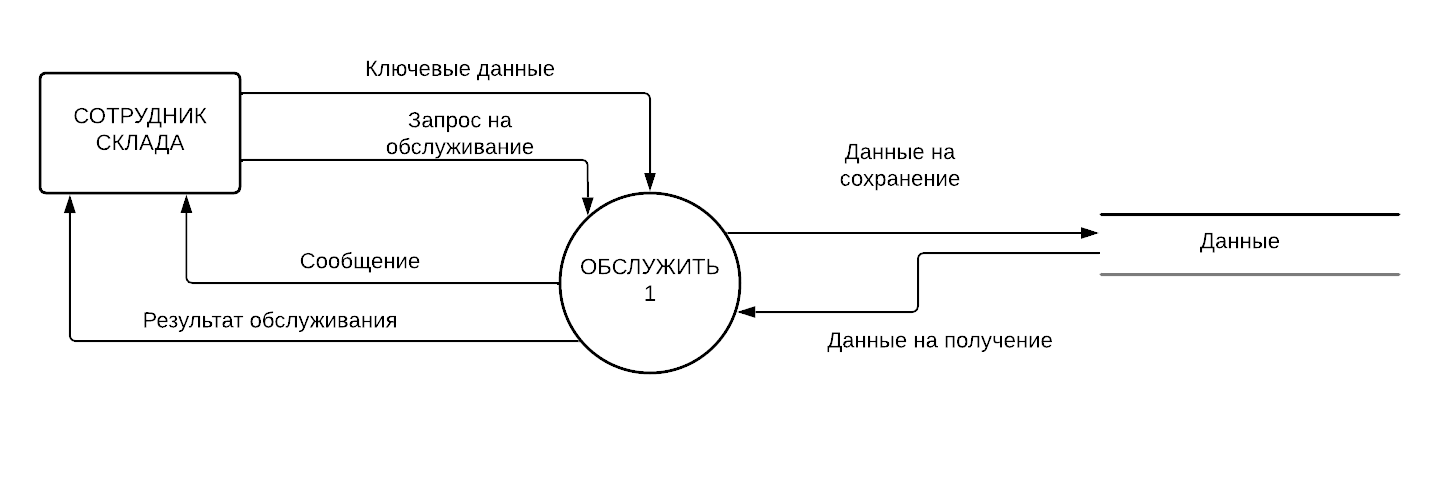


Рис. 5

Приведем описание потоков данных, которым обменивается система с внешними сущностями.

Для работы со складской системой сотруднику необходимо авторизоваться в системе, введя свои КЛЮЧЕВЫЕ ДАННЫЕ, т.е логин и пароль, а также сделать ЗАПРОС НА ОБСЛУЖИВАНИЕ, т.е. требуемую ему услугу (получение информации о товаре, добавление или удаление товара, получение данных о поставщиках и т.д.). Обслуживание с позиции сотрудника склада должно обеспечить следующее:

1. Выдать СООБЩЕНИЕ, приглашающее сотрудника ввести КЛЮЧЕВЫЕ ДАННЫЕ и ЗАПРОС НА ОБСЛУЖИВАНИЕ;
2. Выдать клиенту РЕЗУЛЬТАТЫ ОБСЛУЖИВАНИЯ (которые могут включать в себя данные о товарах, информацию о конкретном товаре и информацию об успешности запроса на создание или удаления товара).

Контекстный процесс и БАЗА ДАННЫХ СКЛАДА должны обмениваться следующей информацией:

1. ДАННЫЕ (которые могут включать в себя ДАННЫЕ О ПОСТАВЩИКАХ, ДАННЫЕ О ТОВАРАХ, МЕСТОРАСПОЛОЖЕНИЕ ТОВАРА и т.д.);
2. ПРОТОКОЛ ДЛЯ ОБСЛУЖИВАНИЯ (информация о том, какие данные нужно получить).

Контекстный процесс может быть детализирован DFD первого уровня (рис. 6).

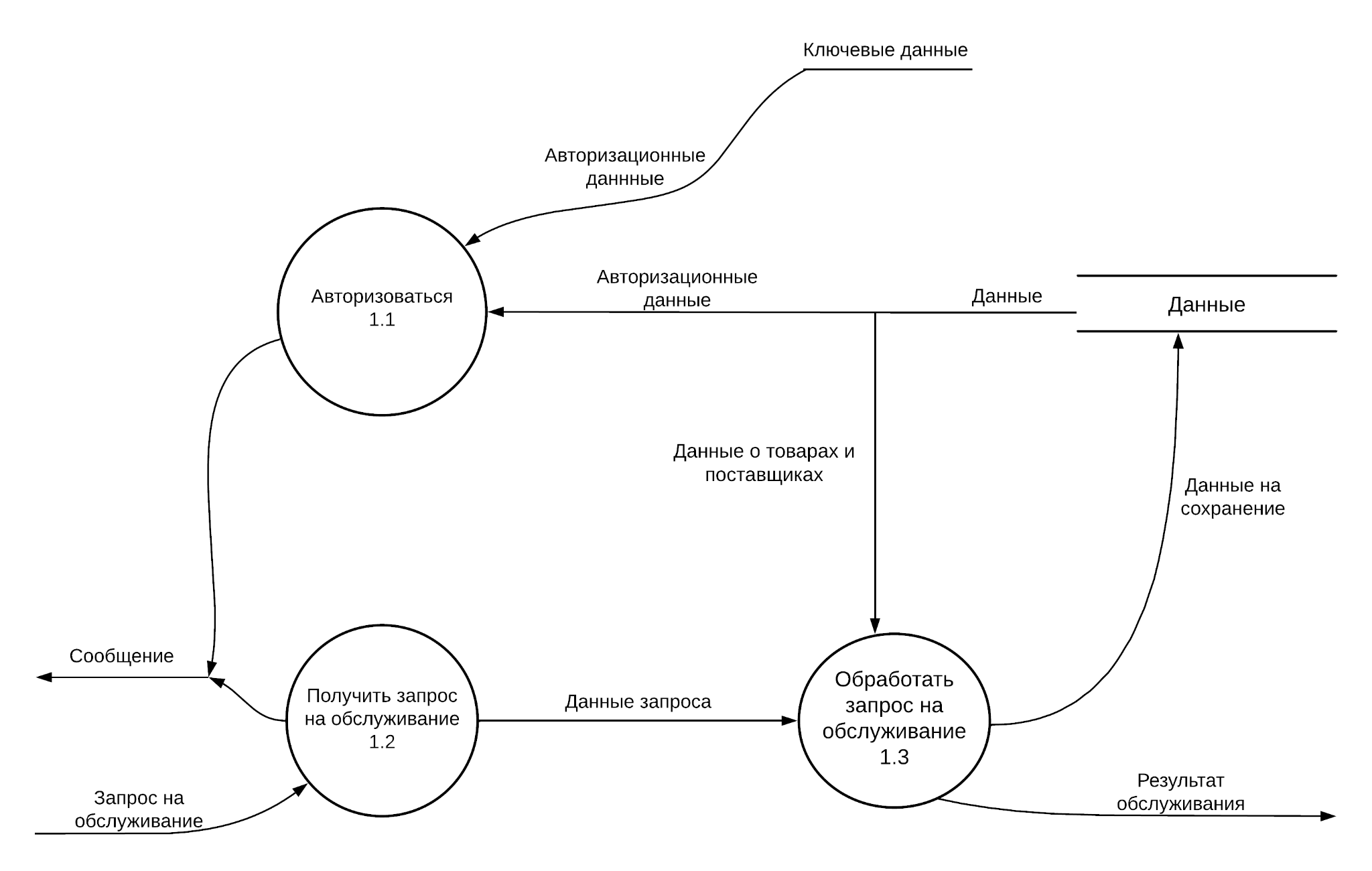


Рис. 6

**Процесс 1.1.** (АВТОРИЗОВАТЬСЯ) осуществляет прием и проверку пароля и логина сотрудника и имеет на входе/выходе следующие потоки:

* внешний выходной поток СООБЩЕНИЕ для информирования клиента о готовности принять пароль;
* входной поток АВТОРИЗАЦИОННЫЕ ДАННЫЕ как элемент внешнего потока КЛЮЧЕВЫЕ ДАННЫЕ;
* входной поток АВТОРИЗАЦИОННЫЕ ДАННЫЕ из хранилища ДАННЫЕ для проверки вводимых пользователем логина и пароля;

**Процесс 1.2.** (ПОЛУЧИТЬ ЗАПРОС НА ОБСЛУЖИВАНИЕ) осуществляет прием и проверку запроса клиента на проведение необходимой ему операции и имеет на входе/выходе следующие потоки:

* внешний выходной поток СООБЩЕНИЕ для информирования клиента о своей готовности принять запрос на обслуживание;
* входной поток ЗАПРОС НА ОБСЛУЖИВАНИЕ;
* выходной поток ДАННЫЕ ЗАПРОСА.

**Процесс 1.3.** (ОБРАБОТАТЬ ЗАПРОС НА ОБСЛУЖИВАНИЕ) осуществляет прием на проверку запроса от работника склада на проведение необходимой ему операции, выдачу результатов этой операции и имеет на входе/выходе следующие потоки:

* входной поток ДАННЫЕ ЗАПРОСА;
* входной поток ДАННЫЕ О ТОВАРАХ И ПОСТАВЩИКАХ из хранилища ДАННЫЕ;
* выходной поток ДАННЫЕ НА СОХРАНЕНИЕ в хранилище ДАННЫЕ;
* выходной поток РЕЗУЛЬТАТ ОБСЛУЖИВАНИЯ.

Процессы 1.1 и 1.3 требуют детализации с помощью DFD второго уровня как показано на рис. 7 и рис. 8 соответственно.

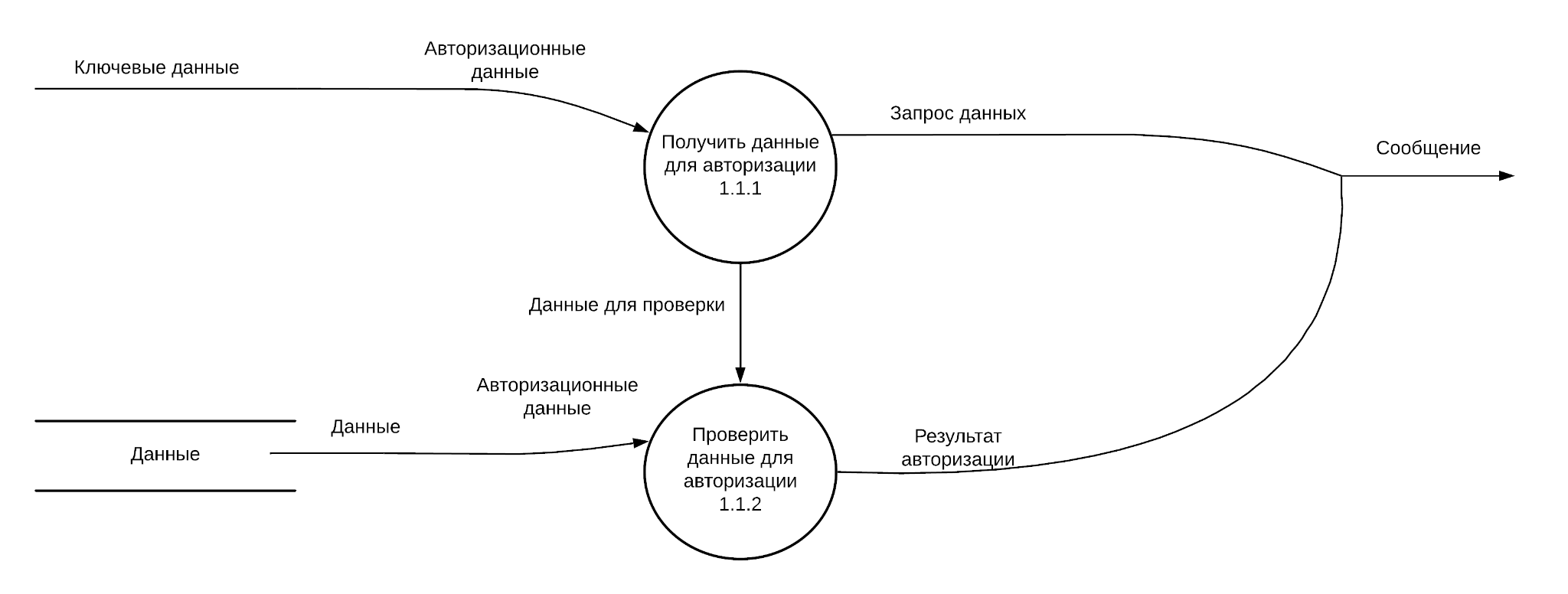


Рис. 8

**Процесс 1.1.1.** (ПОЛУЧИТЬ ДАННЫЕ ДЛЯ АВТОРИЗАЦИИ) необходим для получения данных от пользователя и дальнейшей передачи этих данных в другой процесс:

* выходной поток ЗАПРОС ДАННЫХ, как элемент внешнего потока СООБЩЕНИЕ для информирования клиента о своей готовности принять данные;
* входной поток АВТОРИЗАЦИОННЫЕ ДАННЫЕ, как элемент внешнего потока КЛЮЧЕВЫЕ ДАННЫЕ, которые введет пользователь через форму;
* выходной поток ДАННЫЕ ДЛЯ ПРОВЕРКИ для передачи полученных данных в процесс ПРОВЕРИТЬ ДАННЫЕ ДЛЯ АВТОРИЗАЦИИ.

**Процесс 1.1.2.** (ПРОВЕРИТЬ ДАННЫЕ ДЛЯ АВТОРИЗАЦИИ) для проверки полученных данных и отправке результата пользователю об успешности авторизации:

* входной поток ДАННЫЕ ДЛЯ ПРОВЕРКИ служит для получения данных от процесса ПОЛУЧИТЬ ДАННЫЕ ДЛЯ АВТОРИЗАЦИИ;
* входной поток АВТОРИЗАЦИОННЫЕ ДАННЫЕ, как часть внешнего потока ДАННЫЕ служит для получения авторизационных данных из хранилища;
* выходной поток РЕЗУЛЬТАТ АВТОРИЗАЦИИ, как часть внешнего потока СООБЩЕНИЕ служит для информирования пользователя об успешности авторизации.

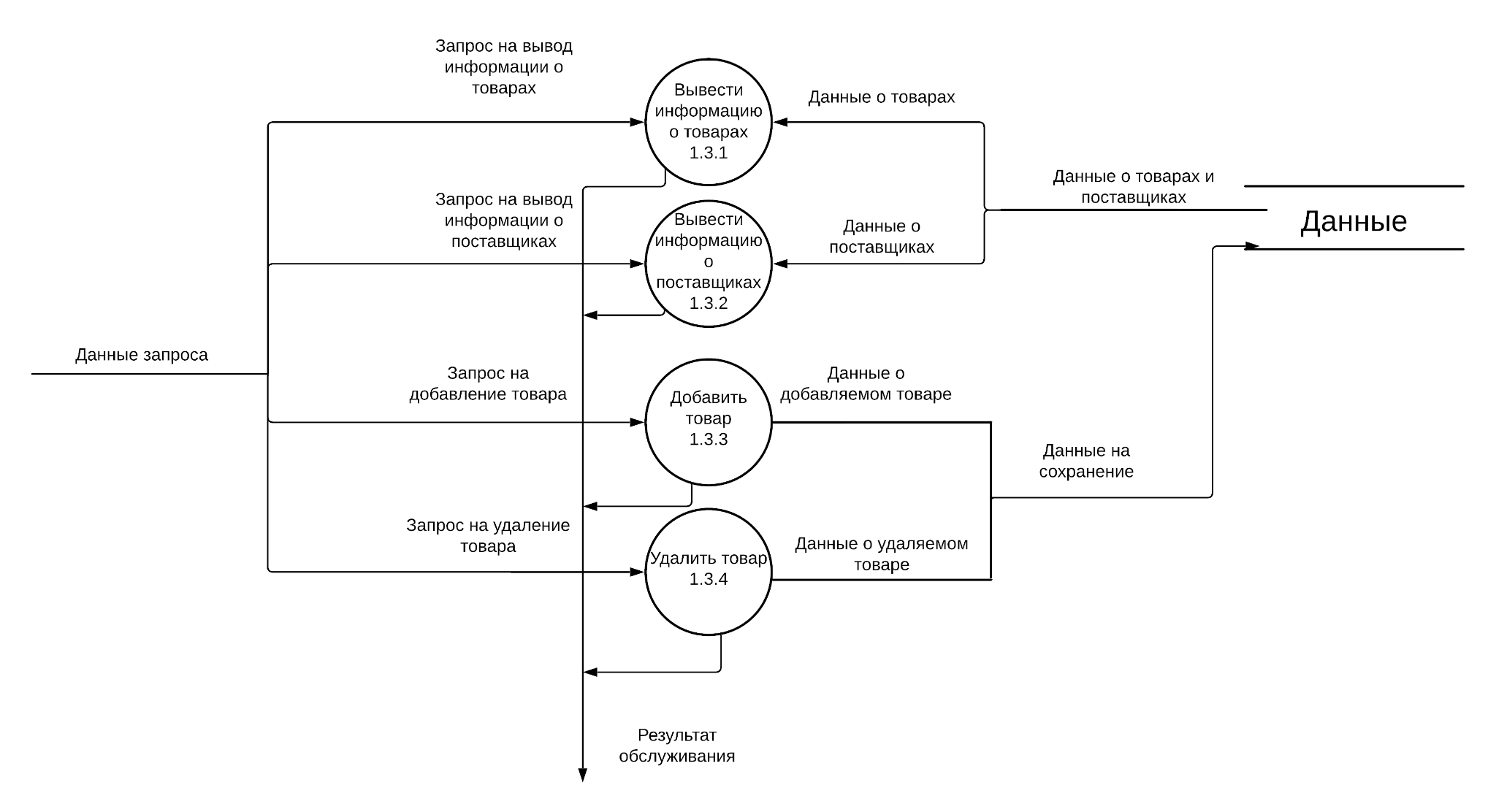


Рис. 8

**Процесс 1.3.1.** (ВЫВЕСТИ ИНФОРМАЦИЮ О ТОВАРАХ) необходим для вывода информации о имеющихся товарах на складе:

* входной поток ЗАПРОС НА ВЫВОД ИНФОРМАЦИИ О ТОВАРАХ, как элемент внешнего потока ДАННЫЕ ЗАПРОСА для получения информации о том, какие именно и в каком количестве выводить товары;
* входной поток ДАННЫЕ О ТОВАРАХ, как элемент внешнего потока ДАННЫЕ О ТОВАРАХ И ПОСТАВЩИКАХ, которые мы получаем из хранилища ДАННЫЕ;
* выходной поток РЕЗУЛЬТАТ ОБСЛУЖИВАНИЯ для получения обработанных данных и передачи этой информации пользователю.

**Процесс 1.3.2.** (ВЫВЕСТИ ИНФОРМАЦИЮ О ПОСТАВЩИКАХ) необходим для вывода информации о поставщиках:

* входной поток ЗАПРОС НА ВЫВОД ИНФОРМАЦИИ О ПОСТАВЩИКАХ, как элемент внешнего потока ДАННЫЕ ЗАПРОСА для получения информации о том, каких именно и в каком количестве выводить поставщиков;
* входной поток ДАННЫЕ О ПОСТАВЩИКАХ, как элемент внешнего потока ДАННЫЕ О ТОВАРАХ И ПОСТАВЩИКАХ, которые мы получаем из хранилища ДАННЫЕ;
* выходной поток РЕЗУЛЬТАТ ОБСЛУЖИВАНИЯ для получения обработанных данных и передачи этой информации пользователю.

**Процесс 1.3.3.** (ДОБАВИТЬ ТОВАР) необходим для добавления товаров сотрудником склада:

* входной поток ЗАПРОС НА ДОБАВЛЕНИЕ ТОВАРА, как элемент внешнего потока ДАННЫЕ ЗАПРОСА служит для получения от сотрудника склада информации о том, какой именно товар нужно создать;
* выходной поток ДАННЫЕ О ДОБАВЛЯЕМОМ ТОВАРЕ, как элемент внешнего потока ДАННЫЕ НА СОХРАНЕНИЕ, которые сохраняются в хранилище ДАННЫЕ;
* выходной поток РЕЗУЛЬТАТ ОБСЛУЖИВАНИЯ для получения статуса успешности выполненной операции по добавлению товара и передачи этой информации пользователю.

**Процесс 1.3.4.** (УДАЛИТЬ ТОВАР) необходим для удаления товаров сотрудником склада:

* входной поток ЗАПРОС НА УДАЛЕНИЕ ТОВАРА, как элемент внешнего потока ДАННЫЕ ЗАПРОСА служит для получения от сотрудника склада информации о том, какой именно товар нужно удалить;
* выходной поток ДАННЫЕ О УДАЛЯЕМОМ ТОВАРЕ, как элемент внешнего потока ДАННЫЕ НА СОХРАНЕНИЕ, которые удаляют определенный товар или их группу в хранилище ДАННЫЕ;
* выходной поток РЕЗУЛЬТАТ ОБСЛУЖИВАНИЯ для получения статуса успешности выполненной операции по удалению товара и передачи этой информации пользователю.

Данные, находящиеся в базе данных, представлены в форме, которую удобно изобразить в виде ER-диаграммы (рис. 9).

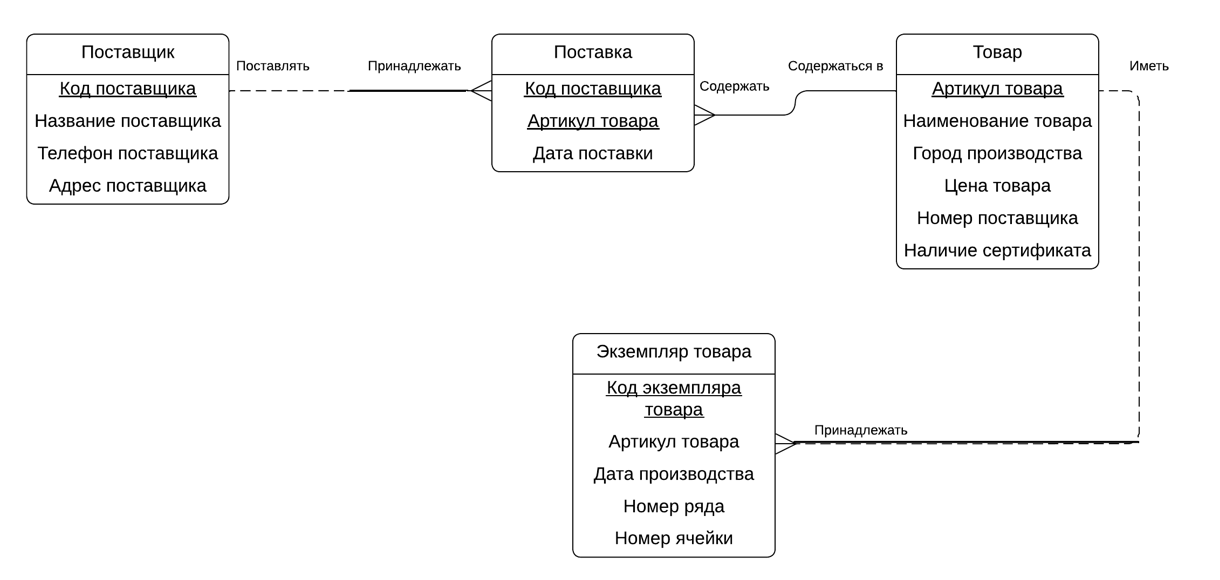


Рис. 9