

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Факультет «Информатика и вычислительная т	техника»		
Кафедра «Программное обеспечение вычисли	тельной техники и	автоматизирова	нных систем»
	3a	в. кафедрой	ПОВТ и АС
		подпись	<u>Долгов В.В.</u> ФИО
	«_	»	2024Γ.
ПОЯСНИТІ	ЕЛЬНАЯ ЗАПИ	СКА	
к курсовому проекту по дисциплине	азы данных		
на тему Проектирование и реализация б	азы данных склада	запчастей	
Автор проекта (работы)		<u>Волкова Э.</u> Ф.И.О.	Ю.
Направление/специальность, профиль/специа	лизация:	112101	
02.03.03 мод направления Математическое обеспечение и наимен	администрировани ование профиля (специали		ных систем
Обозначение курсового проекта (работы)	KP.350000.000	Группа ВМО	31
Руководитель проекта:	ст. препод	цаватель Новик	ов С.П.
подпись		(должность, ФИО)	
Проект (работа) защищен (а)			полинен



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Факультет «Информатика и вычислительная технин	Ka»	
Кафедра <u>«Программное обеспечение вычислительн</u>	пой техники и автоматизи	рованных систем»
	Зав. кафедрой	ПОВТ и АС
	(72,7774)	<u>Долгов В.В.</u>
		2024г.
ЗАДА	ние	
к курсовой работе по дисциплине	Базы данных	
Студент Волкова Э.Ю. КР.350000.000	Группа ВМО31	
Тема Проектирование и реализация базы данны:	х склада запчастей	
Срок представления проекта (работы) к защите «_	»2024г.	
Исходные данные для курсового проекта (работы)		
Задание на выполнение курсовой работы		
Дейт, Кристофер – Введение в системы баз данных, 8		М.: Издательский
дом «Вильяме», 2005 – 1328 с.: ил. – Парал. тит. англ		
Паттон Джефф. Пользовательские истории. Искусс	<u>гво гибкой разработки ПС</u>	<u> О. – СПб.: Питер,</u>
<u>2019. − 288 c.</u>		
Карпова Т.С. Базы данных: модели, разработка, реали	изания. – СПб.: Питер, 200	1 304 c.

Содержание пояснительной записки

ВВЕДЕНИЕ:

В данном разделе содержится: описание ключевых понятий, которые фигурируют в контексте баз данных, постановка задачи, выполнение которой сводится к

разработке базы данных для склада запчастей, краткие описания каждой главы.

Наименование и содержание разделов:

1. Анализ предметной области и проектирование структуры базы данных. Этот

раздел посвящен описанию выбранной предметной области. Приведено

подробное описание видения базы данных конечными пользователями, описаны

их основные функции. Описана модель уровня представлений.

2. Физическая реализация разработанной модели. В данном разделе описаны

основные функции, используемые для работы с базой данных, спроектированы

таблицы в системе управления базой данных.

3. Разработка клиентского интерфейса для работы с базой данных. Данный раздел

содержит описание классов, необходимых для создания приложения, их методов.

Произведено тестирование приложения.

ЗАКЛЮЧЕНИЕ:

В заключении курсовой работы отображены результаты решения поставленной задачи — была разработана база данных склада запчастей, а также программное средство с графическим интерфейсом в виде оконного приложения, соответствующего требованиям, предъявляемыми конечными пользователями.

Руководитель проекта (работы)		Новиков С.П.
• • • • • • • • • • • • • • • • • • • •	подпись, дата	ФИО
-		
Задание принял(а) к исполнению		Волкова Э.Ю.
- · · · · · -	полнись лата	ФИО

Содержание

Введение
1 Анализ предметной области и проектирования структуры базы данных 7
1.1 Формирование уровня представлений базы данных склада запчастей 7
1.1.1 Описание представлений пользователей и выделение основных сущностей
1.1.2 Описание объектов предметной области, их атрибутов и доменов 10
1.1.3 Создание схемы в виде ER-диаграммы
1.2.2 Нормализация отношений
1.2.2.1 Нормализация до первой нормальной формы (1H Φ)
1.2.2.2 Выбор первичных ключей и нормализация до второй нормальной формы (2HФ)
1.2.2.3 Нормализация до третьей нормальной формы (3H Φ)
1.2.3 Логическое (даталогическое) проектирование базы данных
2 Физическая реализация модели БД склада запчастей
2.1 Обоснование выбора системы управления базами данных (СУБД) 25
2.2 Создание таблиц данных в среде целевой СУБД
3 Разработка клиентского интерфейса для работы с базой данных
3.1 Выбор среды для разработки приложения
3.2 Описание основных модулей и функций интерфейса
3.3 Пример работы графического интерфейса
Заключение
Приложение A ER-диаграмма
Приложение Б Даталогическая схема
Приложение В UML-диаграмма классов программного средства
Приложение Г Исходный код программного средства
Приложение Д Исходный код базы данных склада запчастей
Приложение Е Отзыв руководителя

					KP.3500	00	0.0	00	
Изм.	Лист	№ докум.	Подпись	Дата				_	
Разра	ботал	Волкова Э.Ю.				J	Тит.	Лист	Листов
Прове	рил	Новиков С.П.			Проектирование и реализация			4	119
					базы данных склада ДГТУ запчастей кафедра «ПОВ		V		
							radio		
Утвер	одил	Долгов В.В.					кафеора «ПОВТИАС»		

Введение

Каждый день генерируются гигантские объемы данных, работать с которыми без систем хранения и структурирования не предоставляется возможным. Для работы с такими масштабами получаемой информации необходимы инструменты, организующие работу, взаимодействие, хранение и доступ к данным. Таким инструментом и являются базы данных.

База данных — это информационная модель, позволяющая упорядоченном виде хранить данные о группе объектов, обладающих одинаковым набором свойств [1]. Иначе говоря, база данных – это именованная совокупность взаимосвязанных данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, при такой минимальной избыточности, которая допускает ИХ использование оптимальным образом для одного или нескольких приложений. База данных состоит из множества связанных файлов и используется в целях отображения и удовлетворения информационных изменяющегося реального мира потребностей пользователей. Данные запоминаются так, чтобы они были независимы от программ, использующих эти данные. Для добавления новых или модификации существующих данных, а также для поиска данных в базе применяется система управления базами данных (СУБД).

СУБД — это программное обеспечение, предназначенное для создания, ведения и совместного использования базы данных многими пользователями. СУБД осуществляют ввод, проверку, систематизацию, поиск и обработку данных, распечатку их в виде отчётов. Наиболее распространенными СУБД являются MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Главным источником данных при проектировании базы данных является соответствующая предметная область, которая будет рассматриваться как совокупность знаний и данных об объектах и процессах, подлежащих проектированию и хранению в БД.

Изм.	Лист	№ докум.	Подпись	Дата

Целью данной работы является разработка базы данных и графического клиентского интерфейса для склада запчастей.

В первой главе выполнен анализ предметной области, выделены основные сущности и их атрибуты, описана база данных на внешнем и концептуальном уровнях.

Во второй главе описана реализация базы данных склада запчастей на физическом уровне.

В третьей главе описана реализация клиентского интерфейса и рассмотрена работа приложения на тестовом примере.

	·			
Изм.	Лист	№ докум.	Подпись	Дата

1 Анализ предметной области и проектирования структуры базы данных

Проектирование базы данных принято разбивать на несколько уровней абстракций, чтобы точно описать необходимые элементы не только для хранения информации, но и для её обработки.

1.1 Формирование уровня представлений базы данных склада запчастей

1.1.1 Описание представлений пользователей и выделение основных сущностей

Предметная область базы данных для учета запчастей на складе включает в себя информацию о различных аспектах складского хозяйства. Она включает в себя несколько ключевых элементов для базы данных.

Детали: в базе данных должна быть информация о всех деталях, которые хранятся на складе. Каждая запись о детали должна содержать уникальный идентификатор, название, параметры (например, вес, размеры).

Складские помещения и полки: для эффективного учета деталей необходимо иметь информацию о расположении товаров на складе. Это включает в себя данные о складских помещениях и полках, на которых хранятся детали.

Отправки и поставки: база данных должна содержать информацию об отправках, поставках, дате и времени прихода или расхода, количестве деталей, статусе заказа и других связанных с заказами данных.

Учет перемещений: для отслеживания перемещения деталей на складе необходима информация о перемещениях товаров между складскими помещениями, состоянии товаров, ответственных сотрудниках, дате перемещения и других связанных с перемещением данных.

Пользователи и права доступа: для обеспечения безопасности и контроля доступа к данным в базе должна быть реализована система

Изм.	Лист	№ докум.	Подпись	Дата

управления пользователями с различными уровнями доступа к информации в зависимости от их ролей на складе.

База данных склада запчастей необходима прежде всего для хранения информации о пользователях, находящихся на складе деталей, поступивших и выполненных накладных.

Для каждого пользователя должен быть строгий раздел доступных действий, которые может выполнить тот или иной пользователь. Иначе говоря, кладовщик не должен иметь доступ к добавлению новых накладных в базу данных, так как его задача собрать необходимые детали на отгрузку или разложить по полкам поступившие. Накладными должен заниматься менеджер склада, который также будет давать ограниченный доступ к определённым накладным кладовщику. Именно менеджер должен отслеживать этапы отгрузки запчастей со склада или их поступления. Владелец склада должен в свою очередь добавлять новых пользователей, выдавать им роль в базе данных склада, а также добавлять новых контрагентов, с которыми он сотрудничает.

На основе предметной области можно выделить следующих пользователей: кладовщик, менеджер склада, владелец склада.

Кладовщик: осуществляет приемку поставок, выкладку товаров на полки, при этом происходит заполнение в таблице «Стеллаж», указывает данные в таблице «Полка», поле чего формируется ссылки на «Стеллаж», «Помещение» и «Склад». При приёме поставки к соответствующей накладной прикрепляется сотрудник с ролью Кладовщик. Данный сотрудник получает в виде документа список деталей, а также их местоположение на складе, которое предварительно сформировано с помощью запроса к таблицам: «Склад», «Помещение», «Стеллаж», «Полка», «Деталь». Сотрудник также может изменять поле «Состояние» (завершено/в процессе) отображаемого таблице соответствии c выполнением задачи, «Накладная». После внесения изменений проверяется количество имеющихся деталей, при установке статуса «завершено», количество деталей

Изм.	Лист	№ докум.	Подпись	Дата

на складе уменьшается/увеличивается, полки освобождаются/заполняются, происходит изменения в таблицах: «Склад», «Помещение», «Стеллаж», «Полка», «Деталь».

Кладовщику важно знать, где именно на складе находится определённая деталь, поэтому необходимо реализовать поиск, по следующим таблицам: Склад, Помещение, Стеллаж, Полка, Деталь. Таким образом можно выделить следующие сущности: «Склад», «Помещение», «Стеллаж», «Полка», «Деталь», «Накладная».

Менеджер склада: получает в физическом виде накладные на выгрузку деталей на склад и на отгрузку деталей со склада. Сотрудник с такой ролью заполняет таблицу «Накладные», где указывает номер договора с контрагентом (табл.), дату и время выгрузки или отгрузки деталей со склада, а также статус выгрузки или отгрузки, список деталей, и закреплённого за накладного кладовщика.

Таким образом менеджеру склада необходимо вести учёт накладных на поставку и отправку деталей, что выделяется в следующие таблицы: «Накладные», «Накладные(сотрудник)», «Накладные(деталь)», «Контрагент», «Сотрудник».

Владелец склада: ему необходимо отслеживать информацию о сотрудниках, работающих на данном складе, а также о контрагентах, с которыми ведётся сотрудничество. Сотрудник с данной ролью может вносить изменения в таблицу «Сотрудник», заполняя (изменяя) поля «Фамилия», «Имя», «Отчество», «Роль». Владелец склада также должен иметь доступ к таблице «Контрагент», в которую он может вносить изменения: добавление новой записи о контрагенте, изменение уже имеющейся записи, удаление записи. В данных записях владелец склада заполняет (изменяет) поля «Наименование», «Контактное лицо», «Номер телефона» и «Адрес» (полный адрес, т.е. область, город, улица и т.д.).

Таким образом можно выделить следующие сущности: «Сотрудник» и «Контрагент».

Изм.	Лист	№ докум.	Подпись	Дата

Итого выделено десять сущностей: «Склад», «Помещение», «Стеллаж», «Полка», «Деталь», «Накладные», «Накладные(сотрудник)», «Накладные(деталь)», «Сотрудник», «Контрагент».

1.1.2 Описание объектов предметной области, их атрибутов и доменов

Далее будут описаны необходимые сущности и атрибуты для создания базы данных склада запчастей.

Таблица 1.1 – Отношение «warehouse» к сущности «Склад»

Имя атрибута	Домен атрибута
warehouse_id	Серийный номер
warehouse_number	Целое число
address	Текст

Ограничения:

warehouse_id – первичный ключ (уникальные значения).

Таблица 1.2 – Отношение «room» к сущности «Помещение»

Имя атрибута	Домен атрибута
room_id	Серийный номер
warehouseID	Серийный номер
room_number	Целое число

Ограничения:

room_id – первичный ключ (уникальные значения);

warehouseID – внешний ключ, ссылающийся на поле warehouse_id в таблице «Склад».

Таблица 1.3 – Отношение «rack» к сущности «Стеллаж»

Имя атрибута	Домен атрибута
rack_id	Серийный номер
roomID	Серийный номер
rack_number	Целое число

Ограничения:

rack_id – первичный ключ (уникальные значения);

Изм.	Лист	№ докум.	Подпись	Дата

roomID – внешний ключ, ссылающийся на поле rack_id в таблице «Помещение».

Таблица 1.4 – Отношение «shelf» к сущности «Полка»

Имя атрибута	Домен атрибута
shelf_id	Серийный номер
rackID	Серийный номер
shelf_number	Целое число

Ограничения:

shelf_id – первичный ключ (уникальные значения);

rackID – внешний ключ, ссылающийся на поле rack_id в таблице «Стеллаж».

Таблица 1.5 – Отношение «details» к сущности «Деталь»

Имя атрибута	Домен атрибута
detail_id	Серийный номер
shelfID	Серийный номер
weight	Число с плавающей точкой
type_detail	Текст

Ограничения:

detail_id – первичный ключ (уникальные значения);

shelfD – внешний ключ, ссылающийся на поле shelf_id в таблице «Полка».

Таблица 1.6 – Отношение «counteragent» к сущности «Контрагент»

Имя атрибута	Домен атрибута
counteragent_id	Серийный номер
counteragent_name	Строка (128)
contact_person	Строка (128)
phone_number	Большое целое число
address	Текст

Ограничения:

counteragent_id – первичный ключ (уникальные значения).

Изм.	Лист	№ докум.	Подпись	Дата

Таблица 1.7 – Отношение «invoice» к сущности «Накладная»

Имя атрибута	Домен атрибута
invoice_id	Серийный номер
counteragentID	Серийный номер
date_time	Дата и время
type_invoice	Булево значение
status	Булево значение

Ограничения:

counteragent_id – первичный ключ (уникальные значения);

counteragentID – внешний ключ, ссылающийся на поле counteragent_id в таблице «Контрагент».

Таблица 1.8 – Отношение «invoice_detail» к сущности «Накладная(деталь)»

Имя атрибута	Домен атрибута
invoiceID	Серийный номер
detailID	Серийный номер
quantity	Целое число

Ограничения:

invoice_id – первичный ключ (уникальные значения);

detailID – внешний ключ, ссылающийся на поле counteragent_id в таблице «Деталь».

Таблица 1.9 – Отношение «employee» к сущности «Сотрудник»

Имя атрибута	Домен атрибута
employee_id	Серийный номер
employee_role	Строка (25)
last_name	Строка (35)
first_name	Строка (35)
patronymic	Строка (35)

Ограничения:

employee_id – первичный ключ (уникальные значения).

Изм.	Лист	№ докум.	Подпись	Дата

Таблица 1.10 – Отношение «invoice_employee» к сущности «Накладная(сотрудник)»

Имя атрибута	Домен атрибута
invoiceID	Серийный номер
responsible	Серийный номер
granted_access	Серийный номер
when_granted	Дата и время

Ограничения:

invoiceID – внешний ключ, ссылающийся на поле invoice_id в таблице «Накладная»;

responsible — внешний ключ, ссылающийся на поле employee_id в таблице «Сотрудник»;

granted_access – внешний ключ, ссылающийся на поле employee_id в таблице «Сотрудник».

1.1.3 Создание схемы в виде ER-диаграммы

Создание логической схемы является следующим этапом в процессе разработки и проектирования базы данных магазина.

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

В процессе разработки логическая модель данных постоянно тестируется и проверяется на соответствие требованиям пользователей.

Такая модель данных является источником информации для этапа физического проектирования и обеспечивает разработчика физической базы данных средствами поиска компромиссов, необходимых для достижения поставленных целей. При правильной организации сопровождения поддерживаемая модель данных позволяет точно и наглядно представить

Изм.	Лист	№ докум.	Подпись	Дата

вносимые в базу данных изменения, а также оценить их влияние на прикладные программы и использование данных, уже имеющихся в базе. В настоящий момент фактическим стандартом в моделировании баз данных стала модель «сущность-связь».

Рассмотрим основные понятия, используемые для построения ERдиаграммы.

Сущность — это реальный или представляемый тип объекта, информация о котором должна сохраняться и быть доступна. В диаграммах сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности — это имя типа, а не некоторого конкретного экземпляра этого типа. Каждый экземпляр сущности (объект) должен быть отличим от любого другого экземпляра той же сущности.

Связь — это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Связь может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). Возможны связи на основе отношений: один-к-одному, один-ко-многим, многие-к-одному, многие-ко-многим.

Один-к-одному предполагает, что в каждый момент времени каждому элементу (кортежу) А соответствует 0 или 1 элементов (кортежей) В.

Один-ко-многим состоит в том, что в каждый момент времени каждому элементу (кортежу) А соответствует несколько элементов (кортежей) В.

Многие-к-одному предполагает, что в каждый момент времени множеству элементов (кортежей) A соответствует 1 элемент (кортеж) В.

Многие-ко-многим состоит в том, что в каждый момент времени множеству элементов (кортежей) А соответствует множество элементов (кортежей) В [2].

Таким образом, на основе представлений пользователей базы данных можно составить обобщенное представление предметной области, выделив основные объекты будущей базы данных и отбросив второстепенные. В качестве концептуальной модели была выбрана ER-диаграмма. ER-

Изм.	Лист	№ докум.	Подпись	Дата

диаграммы удобны тем, что процесс выделения сущностей, атрибутов и связей является цикличным. ER-модель использует графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями.

В базе данных «Склад запчастей» можно выделить следующие основные сущности: «Сотрудник», «Контрагент», «Склад», «Помещение», «Стеллаж», «Полка», «Накладная», «Накладная(деталь)», «Накладная(сотрудник)», «Деталь».

Рассмотрим атрибуты каждой сущности, необходимые конечным пользователям для работы.

«Сотрудник»: фамилия, имя, отчество, роль.

«Контрагент»: наименование, контактное лицо, номер телефона, адрес.

«Склад»: номер, адрес.

«Помещение»: номер.

«Стеллаж»: номер.

«Полка»: номер.

«Накладная»: номер договора, дата/время, тип (загрузка/отгрузка), статус.

«Накладная(деталь)»: серийный номер, количество, номер накладной.

«Накладная(сотрудник)»: ответственное лицо, номер накладной.

«Деталь»: серийный номер, вид детали, вес.

Основные связи в проектируемой ER-диаграмме: каждая «Деталь» находится на «Полке», которая находится в «Стеллаже». «Стеллаж» находится в «Помещении», а оно находится на «Складе». «Накладная» оформляется «Контрагентом». «Сотрудник» имеет доступ к «Накладной(сотрудник)», которая включает в себя «Накладную». «Накладной(деталь)» включает себя «Деталь» и «Накладную».

Таким образом, на основе приведенных представлений пользователей, выделенных основных объектов, описанных атрибутов и связей была

Изм.	Лист	№ докум.	Подпись	Дата

спроектирована семантическая модель предметной области склада запчастей (Приложение A: ER-диаграмма).

1.1.4 Описание связей между объектами

- 1. Склад (1) Помещение (N): в таблице Склад каждая запись связана со множеством помещений из таблицы Помещение. Это связь одинко-многим, так как к одному складу привязано множество помещений, но каждое помещение находится только на одном складе;
- 2. Помещение (1) Стеллаж (N): в таблице Помещение каждая запись связана с множеством стеллажей из таблицы Стеллаж. Это связь одинко-многим, так как к одному помещению привязано множество стеллажей, но каждый стеллаж находится только в одном помещении;
- 3. Стеллаж (1) Полка (N): в таблице Стеллаж каждая запись связана со множеством полок из таблицы Полка. Это связь один-ко-многим, так как к одному стеллажу привязано множество полок, но каждая полка находится только на одном стеллаже;
- 4. Полка (1) Деталь (N): в таблице Полка каждая запись связана со множеством деталей из таблицы Деталь. Это связь один-ко-многим, так как к одной полке привязано множество деталей, но каждая деталь находится только на одной полке;
- 5. Накладная(деталь) (N) Накладная(сотрудник) (N): в таблице Накладная(деталь) каждая запись связана с каждым сотрудником ответственным за выполнение отгрузки/загрузки деталей с/на склад из таблицы Накладная(сотрудник). Это связь многие-ко-многим, так как множество записей таблицы Накладная(детали) связаны со множеством записей таблицы Накладная(сотрудник). Такое отношение стоит разделить на две части, а именно на связь многие-к-одному (Накладная(деталь) (N) Накладная (1)) и один-ко-многим (Накладная (1) Накладная(сотрудник) (N));

Изм.	Лист	№ докум.	Подпись	Дата

- 6. Накладная(деталь) (N) Накладная (1): в таблице Накладная(деталь) каждая запись связана с одной накладной из таблицы Накладная. Это связь многие-к-одному, так как каждая накладная с количеством определённой детали привязана только к одной накладной, но каждая накладная может иметь несколько накладных с количеством определённых деталей;
- 7. Накладная (1) Накладная(сотрудник) (N): в таблице Накладная каждая запись связана со множеством накладных доступных определённым сотрудникам из Накладная(сотрудник). Это связь один-ко-многим, так как к одной накладной привязано множество накладных с привязанными сотрудниками, но каждая накладная с привязанными сотрудниками включает в себя только одну накладную;
- 8. Деталь (1) Накладная(деталь) (N): в таблице Деталь каждая запись связана со множеством накладных с количеством определённых деталей из таблицы Накладная(деталь). Это связь один-ко-многим, так как к одной детали привязано множество накладных количеством определённых деталей, но каждая накладная количеством определённых деталей содержит в себе только одну деталь;
- 9. Накладная(сотрудник) (N) Сотрудник (1): в таблице Накладная(сотрудник) каждая запись связана с одним сотрудником из таблицы Сотрудник. Это связь многие-к-одному, так как каждая накладная с привязанными сотрудниками привязана только к одному сотруднику, но каждый сотрудник может быть привязан к нескольким накладным;
- 10. Накладная (N) Контрагент (1): в таблице Накладная каждая запись связана с одним контрагентом из таблицы Контрагент. Это связь многие-к-одному, так как каждая накладная привязана только к одному контрагенту, но каждый контрагент может оформлять несколько накладных.

Изм.	Лист	№ докум.	Подпись	Дата

1.2 Построение концептуального уровня базы данных

1.2.1 Описание функциональных зависимостей, имеющих место в предметной области

warehouse = {warehouse_id, warehouse_number, address}
{warehouse_number} -> {address}

Ограничение: по одному адресу не могут стоять несколько складов.

Каждый склад идентифицируется по уникальному warehouse_id и атрибуту address, так как невозможно чтобы два склада находились по одному адресу. Атрибут address функционально зависит от warehouse_number.

Таблица warehouse имеет в качестве первичного ключа warehouse_id.

room = {room_id, room_number, warehouseID}

{warehouseID, room_number} -> {room_number}

<u>Ограничение:</u> на каждом складе может быть несколько комнат, но номер комнаты должен быть уникальным в пределах одного склада

Каждое помещение идентифицируется по уникальному room_id, первичному ключу, а также по атрибуту warehouseID, который является внешним ключом, ссылающийся на таблицу warehouse. Атрибут room_number функционально зависит от warehouseID и обозначает номер помещения в пределах определённого склада.

rack = {rack_id, rack_number, roomID}
{roomID, rack_number} -> {rack_number}

<u>Ограничение:</u> в каждой комнате может быть несколько стеллажей, но номер стеллажа должен быть уникальным в пределах одной комнаты

Каждый идентифицируется стеллаж ПО уникальному rack id, первичному ключу, а также по атрибуту roomID, который является внешним ссылающийся таблицу Атрибут на room. rack number ключом, функционально зависит от roomID и обозначает номер стеллажа в пределах определённого помещения.

Изм.	Лист	№ докум.	Подпись	Дата

```
shelf = {shelf_id, shelf_number, rackID}
{rackID, shelf_number} -> {shelf_number}
```

<u>Ограничение:</u> в одном стеллаже не могут быть несколько полок с одинаковым номером.

Каждая полка идентифицируется по уникальному shelf_id, первичному ключу, а также по атрибуту rackID, который является внешним ключом, ссылающийся на таблицу rack. Атрибут shelf_number функционально зависит от rackID и обозначает номер полки в пределах определённого стеллажа.

counteragent = {counteragent_id, counteragent_name, contact_person,
phone_number, address}

```
{counteragent_name} -> {contact_person, phone_number}
```

Каждый контрагент идентифицируется по уникальному counteragent_id, первичному ключу. Атрибуты contact_person и phone_number функционально зависят от атрибута counteragent_name и обозначают контактное лицо и его номер, по которому стоит звонить, чтобы связаться с контрагентом.

```
invoice = {invoice_id, counteragentID, date_time, type_invoice, status}
{invoice_id, counteragentID} -> {type_invoice}
```

Каждая накладная идентифицируется по уникальному invoice_id, первичному ключу. Атрибут type_invoice функционально зависит от counteragentID и обозначают тип накладной, созданной определённым контрагентом.

```
invoice_detail = {invoiceID, detailID, quantity}
{invoiceID, detailID} -> {quantity}
```

Каждая накладная идентифицируется по уникальному invoiceID, внешнему ключу, ссылающийся на таблицу invoice, а также по уникальному detailID, также внешнему ключу, который ссылается на таблицу details. Атрибут quantity функционально зависит от них и обозначает количество определённой детали по определённой накладной.

Изм.	Лист	№ докум.	Подпись	Дата

1.2.2 Нормализация отношений

1.2.2.1 Нормализация до первой нормальной формы (1НФ)

Отношение находится в первой нормальной форме (1НФ), если каждый его атрибут атомарен, т.е. содержит только одно значение, а не список значений.

Рассмотрим базовые отношения базы данных «Склад запчастей».

warehouse (1) = {warehouse_id, warehouse_number, address}

room (1) = {room_id, room_number, warehouseID} (с внешним ключом warehouseID)

rack (1) = {rack_id, rack_number, roomID} (с внешним ключом roomID)

shelf (1) = {shelf_id, shelf_number, rackID} (с внешним ключом rackID)

details (1) = {detail_id, shelfID, weight, type_detail} (с внешним ключом shelfID)

counteragent (1) = {counteragent_id, counteragent_name, contact_person,
phone_number}

invoice (1) = {invoice_id, counteragentID, date_time, type_invoice, status} (с внешним ключом counteragentID)

invoice_detail (1) = {invoiceID, detailID, quantity} (составной ключ, состоящий из двух столбцов invoiceID и detailID)

employee (1) = {employee_id, employee_role, last_name, first_name,
patronymic}

invoice_employee (1) = {invoiceID, responsible, granted_access, when_granted} (внешние ключи invoiceID, responsible, granted_access)

Все базовые отношения находятся в 1НФ.

				·
Изм.	Лист	№ докум.	Подпись	Дата

1.2.2.2 Выбор первичных ключей и нормализация до второй нормальной формы (2HФ)

Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме и неключевые атрибуты зависят от первичного ключа. Рассмотрим подробнее таблицы базы данных «Склад запчастей».

warehouse (2) = {warehouse_id, warehouse_number, address}

В качестве первичного ключа был выбран warehouse_id для уникальной идентификации складов в базе данных.

room (2) = {room_id, room_number, warehouseID} (с внешним ключом warehouseID)

В качестве первичного ключа был выбран room_id для уникальной идентификации помещений в базе данных.

rack (2) = {rack_id, rack_number, roomID} (с внешним ключом roomID)

В качестве первичного ключа был выбран rack_id для уникальной идентификации стеллажей в базе данных.

shelf (2) = {shelf_id, shelf_number, rackID} (с внешним ключом rackID)

В качестве первичного ключа был выбран shelf_id для уникальной идентификации полок в базе данных.

details (2) = {detail_id, shelfID, weight, type_detail} (с внешним ключом shelfID)

В качестве первичного ключа был выбран detail_id для уникальной идентификации деталей в базе данных.

counteragent (2) = {counteragent_id, counteragent_name, contact_person,
phone_number}

В качестве первичного ключа был выбран counteragent_id для уникальной идентификации контрагентов в базе данных.

invoice (2) = {invoice_id, counteragentID, date_time, type_invoice, status} (с внешним ключом counteragentID)

Изм.	Лист	№ докум.	Подпись	Дата

В качестве первичного ключа был выбран invoice_id для уникальной идентификации накладных в базе данных.

employee (2) = {employee_id, employee_role, last_name, first_name,
patronymic}

В качестве первичного ключа был выбран employee_id для уникальной идентификации сотрудников в базе данных.

invoice_employee (2) = {invoiceID, responsible, granted_access, when_granted} (внешние ключи invoiceID, responsible, granted_access)

В данной таблице первичным ключом является атрибут invoiceID, который является внешним ключом, ссылающийся на первичный ключ invoice_id из таблицы invoice. Таким образом, неключевые атрибуты данной таблицы будут однозначно зависимы от номера накладной.

invoice_detail (2) = {invoiceID, detailID, quantity} (составной ключ, состоящий из двух столбцов invoiceID и detailID)

В данной таблице первичным ключом, стоит взять составной ключ из атрибутов invoiceID и detailID, которые являются ссылками первичных ключей invoice_id из таблицы invoice и detail_id из таблицы details. Таким образом атрибут quantity однозначно зависит от обеих частей первичного ключа, так как количество деталей зависит от номера самой детали и от номера накладной. Следовательно, база данных приведена ко второй нормальной форме.

1.2.2.3 Нормализация до третьей нормальной формы (ЗНФ)

Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме и между атрибутами нет транзитивности. Рассмотрим подробнее таблицы базы данных «Склад запчастей».

warehouse (3) = {warehouse_id, warehouse_number, address}

Изм.	Лист	№ докум.	Подпись	Дата

Все атрибуты зависят от warehouse_id, транзитивных зависимостей нет.

room (3) = {room_id, room_number, warehouseID} (с внешним ключом warehouseID)

Все атрибуты зависят от room_id.

rack (3) = {rack_id, rack_number, roomID} (с внешним ключом roomID)Bce атрибуты зависят от rack_id.

shelf (3) = {shelf_id, shelf_number, rackID} (с внешним ключом rackID) Все атрибуты зависят от shelf id.

details (3) = {detail_id, shelfID, weight, type_detail} (с внешним ключом shelfID)

Все атрибуты зависят от detail_id.

counteragent (3) = {counteragent_id, counteragent_name, contact_person,
phone_number}

Все атрибуты зависят от counteragent_id.

invoice (3) = {invoice_id, counteragentID, date_time, type_invoice, status}
(c внешним ключом counteragentID)

Все атрибуты зависят от invoice id.

employee (3) = {employee_id, employee_role, last_name, first_name,
patronymic}

Все атрибуты зависят от employee_id.

Таблица **invoice** была разделена на две таблицы: **invoice_employee** и **invoice_detail**, так как в первоначальной таблице существовала транзитивность между атрибутами таблицы. Таким образом, с помощью декомпозиции таблицы **invoice** база данных «Склад запчастей» была приведена к третьей нормальной форме.

invoice_employee (3) = {invoiceID, responsible, granted_access, when_granted} (внешние ключи invoiceID, responsible, granted_access)

Все атрибуты зависят от составного ключа.

invoice_detail (3) = {invoiceID, detailID, quantity} (составной ключ, состоящий из двух столбцов invoiceID и detailID)

Изм.	Лист	№ докум.	Подпись	Дата

Все атрибуты зависят от составного ключа.

В результате проделана нормализация отношения, а структура базы данных «Склад запчастей» приведена к виду, обеспечивающему минимальную логическую избыточность.

1.2.3 Логическое (даталогическое) проектирование базы данных

На этапе логического проектирования происходит разработка логической структуры будущей базы данных «Склад запчастей».

Основой для создания базы данных служит схема базы данных. Инфологическая модель данных, построенная в виде ER-диаграммы, должна быть преобразована в схему БД. Данное преобразование осуществляется путем сопоставления каждой сущности и каждой связи, имеющей атрибуты, с таблицами-отношениями.

В реляционной модели данных даталогическая модель представляет собой набор схем отношений. В ней указываются первичные ключи и связи между отношениями, которые реализованы с помощью внешних ключей.

Ранее были описаны связи между отношениями в базе данных склада запчастей. Результатом логического проектирования является даталогическая схема БД склада запчастей (Приложение Б – Даталогическая схема).

Изм.	Лист	№ докум.	Подпись	Дата

2 Физическая реализация модели БД склада запчастей

2.1 Обоснование выбора системы управления базами данных (СУБД)

В современном мире существует широкий выбор систем управления базами данных (СУБД), каждая из которых обладает своими особенностями.

Oracle Database – это профессиональное решение для разработки сложных приложений. Его основой является высокопроизводительная база данных, способная хранить огромные объемы информации благодаря масштабируемости. Система эффективно обрабатывает запросы множества пользователей без потери производительности даже при резком росте Еще Oracle нагрузки. преимуществом одним является кроссплатформенность. Однако использование этой СУБД требует значительных финансовых затрат на серверное оборудование и техническую поддержку [3].

МуSQL – популярная реляционная СУБД с открытым исходным кодом. В настоящее время разработкой и поддержкой MySQL занимается корпорация Oracle, которая приобрела права на эту систему после поглощения Sun Microsystems (ранее владевшей MySQL AB). Одним из ключевых достоинств MySQL является поддержка различных типов таблиц, включая MyISAM (с полнотекстовым поиском) и InnoDB (с транзакционной обработкой). Также в состав СУБД входит демонстрационный тип таблиц EXAMPLE, иллюстрирующий механизм создания новых форматов данных. Благодаря открытой лицензии (GPL) сообщество разработчиков активно расширяет функционал MySQL, добавляя новые возможности.

PostgreSQL — это объектно-реляционная СУБД с открытым исходным кодом. В отличие от чисто реляционных систем, PostgreSQL предлагает более гибкие и надежные механизмы работы с данными, включая поддержку

Изм.	Лист	№ докум.	Подпись	Дата

их целостности. СУБД распространяется под либеральной BSD-лицензией, что позволяет свободно использовать и модифицировать ее код без юридических ограничений [5].

Для реализации физической модели была выбрана PostgreSQL, поскольку она предоставляет широкий набор инструментов для резервного мониторинга, миграции копирования, данных И администрирования (например, pgAdmin, DBeaver). Это упрощает интеграцию с другими платформами И делает процесс управления базой данных более эффективным.

2.2 Создание таблиц данных в среде целевой СУБД

SQL (язык структурированных запросов) — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. [1]

В среде администрирования и разработки PostgreSQL с помощью SQLзапросов была создана база данных «Warehouse_DB», а также все необходимые таблицы, такие как «warehouse», «room», «rack», «invoice» и другие. Данные SQL-запросы изображены на рисунках 1 и 2.

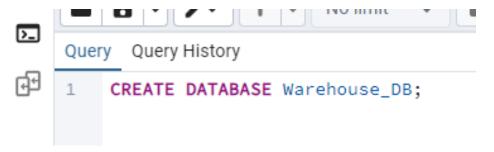


Рисунок 1 – Создание БД «Warehouse_DB»

Изм.	Лист	№ докум.	Подпись	Дата

```
Query Query History
1 		 CREATE TABLE invoice
2
3
        invoice_id serial PRIMARY KEY,
        counteragentID serial NOT NULL,
5
        date_time timestamp NOT NULL,
        type_invoice bool NOT NULL,
6
        status bool NOT NULL,
7
8
        FOREIGN KEY (counteragentID) REFERENCES counteragent (counteragent_id)
9
  );
```

Рисунок 2 – Создание таблицы «invoice»

Для реализации темпоральности в базе данных используется таблица «log_table», которая фиксирует изменения в других таблицах, включая тип изменения (INSERT, UPDATE, DELETE), идентификатор записи и значения до и после изменения. SQL-запрос для создания данной таблицы показан на рисунке 3.

```
CREATE TABLE IF NOT EXISTS public.log_table

(
    log_id integer NOT NULL DEFAULT nextval('log_table_log_id_seq'::regclass),
    table_name character varying(50) COLLATE pg_catalog."default" NOT NULL,
    action_type character varying(20) COLLATE pg_catalog."default" NOT NULL,
    record_id integer,
    action_time timestamp without time zone DEFAULT CURRENT_TIMESTAMP,
    old_values jsonb,
    new_values jsonb,
    CONSTRAINT log_table_pkey PRIMARY KEY (log_id)
)
```

Рисунок 3 – Создание таблицы «log table»

Для автоматизации логирования изменений в таблицах созданы функции и триггеры. Например, функция log_warehouse_changes() записывает информацию об операциях в таблице «warehouse», код которой показан на рисунке 4.

Изм.	Лист	№ докум.	Подпись	Дата

```
CREATE OR REPLACE FUNCTION public.log_warehouse_changes()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
   IF (TG_OP = 'INSERT') THEN
        INSERT INTO log_table (table_name, action_type, record_id, new_values)
        VALUES ('warehouse', 'INSERT', NEW.warehouse_id, to_jsonb(NEW));
        RETURN NEW;
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO log_table (table_name, action_type, record_id, old_values, new_values)
        VALUES ('warehouse', 'UPDATE', OLD.warehouse_id, to_jsonb(OLD), to_jsonb(NEW));
       RETURN NEW;
    ELSIF (TG_OP = 'DELETE') THEN
       INSERT INTO log_table (table_name, action_type, record_id, old_values)
        VALUES ('warehouse', 'DELETE', OLD.warehouse_id, to_jsonb(OLD));
       RETURN OLD:
    END IF;
END;
$BODY$;
```

Рисунок 4 – Функция log_warehouse_changes()

Аналогичные функции созданы для таблиц «room», «rack», «shelf», «invoice», «invoice_employee», «invoice_detail», «details», «counteragent» и «employee». Эти функции вызываются соответствующими триггерами, которые срабатывают после выполнения операций INSERT, UPDATE или DELETE.

Для обеспечения целостности данных при удалении записей создана функция delete_related_data(), которая каскадно удаляет связанные данные. Например, при удалении склада удаляются все связанные комнаты, стеллажи, полки и детали. Этой функции соответствует триггер trg_delete_related_data, который выполняется перед удалением записи в таблице «warehouse». Данная функция изображена на рисунке 5.

	·			·
Изм.	Лист	№ докум.	Подпись	Дата

```
CREATE OR REPLACE FUNCTION public.delete_related_data()
   RETURNS trigger
    LANGUAGE 'plpgsql'
   COST 100
   VOLATILE NOT LEAKPROOF SECURITY DEFINER
BEGIN
    -- Для склада удаляем связанные комнаты, стеллажи, полки и детали
    IF TG TABLE NAME = 'warehouse' THEN
        -- Удаляем детали через отдельный запрос с явными правами
        PERFORM * FROM details WHERE shelfID IN (
           SELECT shelf_id FROM shelf WHERE rackID IN (
               SELECT rack_id FROM rack WHERE roomID IN (
                   SELECT room_id FROM room WHERE warehouseID = OLD.warehouse_id
        ) LIMIT 1:
        DELETE FROM shelf WHERE rackID IN (
           SELECT rack_id FROM rack WHERE roomID IN (
               SELECT room_id FROM room WHERE warehouseID = OLD.warehouse_id
        );
        DELETE FROM rack WHERE roomID IN (
           SELECT room_id FROM room WHERE warehouseID = OLD.warehouse_id
        DELETE FROM room WHERE warehouseID = OLD.warehouse_id;
    -- Для комнаты удаляем связанные стеллажи, полки и детали
    ELSIF TG_TABLE_NAME = 'room' THEN
        PERFORM * FROM details WHERE shelfID IN (
           SELECT shelf id FROM shelf WHERE rackID IN (
               SELECT rack_id FROM rack WHERE roomID = OLD.room_id
        ) LIMIT 1;
        DELETE FROM shelf WHERE rackID IN (
           SELECT rack_id FROM rack WHERE roomID = OLD.room_id
        DELETE FROM rack WHERE roomID = OLD.room_id;
    -- Для стеллажа удаляем связанные полки и детали
    ELSIF TG_TABLE_NAME = 'rack' THEN
        PERFORM * FROM details WHERE shelfID IN (
           SELECT shelf_id FROM shelf WHERE rackID = OLD.rack_id
        ) LIMIT 1:
       DELETE FROM shelf WHERE rackID = OLD.rack_id;
    -- Для полки удаляем связанные детали
    ELSIF TG_TABLE_NAME = 'shelf' THEN
       PERFORM * FROM details WHERE shelfID = OLD.shelf_id LIMIT 1;
    END IF:
    RETURN OLD;
END;
$BODY$:
```

Рисунок 5 – Функция delete_related_data()

Для удобства работы с данными созданы представления «invoice_details_view» и «warehouse_details_view». Представление «invoice_details_view» объединяет информацию о накладных, включая данные о контрагенте, типе накладной, статусе, деталях и ответственных сотрудниках. Данные представления показаны на рисунках 6 и 7.

Изм.	Лист	№ докум.	Подпись	Дата

```
CREATE OR REPLACE VIEW public.invoice_details_view
SELECT inv.invoice_id,
   ca.counteragent_name,
   inv.date time,
       CASE
           WHEN inv.type_invoice THEN 'Bыгрузка'::text
           ELSE 'Отгрузка'::text
       END AS type_invoice_text,
       CASE
           WHEN inv.status THEN 'Завершено'::text
           ELSE 'B npouecce'::text
       END AS status_text,
   det.type_detail,
   invd.quantity,
   emp.last_name AS responsible_last_name,
   emp.first_name AS responsible_first_name,
   emp.patronymic AS responsible_patronymic,
   emp.employee_id AS responsible_id,
   inv.status AS status bool,
   inv.type_invoice AS type_invoice_bool
  FROM invoice inv
    JOIN invoice_detail invd ON inv.invoice_id = invd.invoiceid
    JOIN details det ON invd.detailid = det.detail_id
    JOIN invoice_employee inv_emp ON inv.invoice_id = inv_emp.invoiceid
    JOIN employee emp ON inv_emp.responsible = emp.employee_id
    JOIN counteragent ca ON inv.counteragentid = ca.counteragent_id;
```

Рисунок 6 – Представление «invoice_details_view»

```
CREATE OR REPLACE VIEW public.warehouse_details_view

AS

SELECT w.warehouse_number,
    r.room_number,
    rk.rack_number,
    s.shelf_number,
    d.type_detail,
    d.weight,
    d.detail_id

FROM warehouse w

JOIN room r ON w.warehouse_id = r.warehouseid
    JOIN rack rk ON r.room_id = rk.roomid
    JOIN shelf s ON rk.rack_id = s.rackid
    JOIN details d ON s.shelf_id = d.shelfid;
```

Рисунок 7 – Представление «warehouse_details_view»

Для поддержания актуальности данных в представлении «invoice_details_view» созданы функции insert_invoice_details_view() и delete_invoice_details_view(), которые управляют вставкой и удалением данных в связанных таблицах. Этим функциям соответствуют триггеры, срабатывающие при операциях с представлением.

Представление «warehouse_details_view» предоставляет информацию о деталях на складе, включая их расположение (склад, комната, стеллаж, полка). Для этого представления также созданы функции и триггеры:

Изм.	Лист	№ докум.	Подпись	Дата

insert_into_warehouse_details() для вставки данных, delete_warehouse_details() для удаления и update_warehouse_details_view() для обновления. Эти функции обеспечивают согласованность данных в иерархии склада.

Для преобразования пользовательских данных в формат, удобный для хранения в базе, созданы функции convert_text_to_boolean() и get_employee_id(), что изображены на рисунках 8 и 9.

```
CREATE OR REPLACE FUNCTION public.convert_text_to_boolean(
    text value text.
    field_type text DEFAULT 'status'::text)
   RETURNS boolean
   LANGUAGE 'plpgsql'
   IMMUTABLE PARALLEL UNSAFE
AS $BODY$
BEGIN
    text value := LOWER(TRIM(text value));
    -- Для типа накладной
   IF field_type = 'type' THEN
       RETURN text_value IN ('выгрузка', 'выгрузить', 'отправка', 'true', '1', 'да', 'yes', 'y');
       RETURN text_value IN ('sasepweho', 'rotoso', 'sыполнено', 'done', 'true', '1', 'ga', 'yes', 'y');
   END IF;
END:
$BODY$;
```

Рисунок 8 – Функция convert_text_to_boolean()

```
CREATE OR REPLACE FUNCTION public.get_employee_id(
   p_last_name character varying,
   p_first_name character varying,
   p_patronymic character varying)
   RETURNS integer
   LANGUAGE 'plpgsql'
   COST 100
   VOLATILE SECURITY DEFINER PARALLEL UNSAFE
AS $BODY$
DECLARE
   v_id integer;
BEGIN
   SELECT employee_id INTO v_id
   FROM employee
   WHERE last_name = p_last_name
   AND first_name = p_first_name
   AND patronymic = p_patronymic;
   IF NOT FOUND THEN
       RAISE EXCEPTION 'Сотрудник % % % не найден',
           p_last_name, p_first_name, p_patronymic;
   END IF:
    RETURN v_id;
END;
$BODYS:
```

Рисунок 9 – Функция get_employee_id()

Изм.	Лист	№ докум.	Подпись	Дата

Функция convert_text_to_boolean() преобразует текстовые значения (например, «завершено» или «выгрузка») в булевы значения (TRUE/FALSE), а функция get_employee_id() возвращает идентификатор сотрудника по его ФИО. Эти функции используются в триггерах и других функциях для автоматизации обработки данных.

Описанные в данном разделе функции и триггеры автоматизируют различные операции с таблицами базы данных склада запчастей и связанными данными, упрощая управление базой данных.

При помощи СУБД PostgreSQL созданы все необходимые таблицы для реализации БД склада запчастей, определены первичные ключи, проведены связи между таблицами, а также созданы таблицы и триггеры для осуществления темпоральности базы данных. Все таблицы были заполнены необходимыми данными.

Исходный код базы данных склада запчастей представлен в Приложении Д.

	·			
Изм.	Лист	№ докум.	Подпись	Дата

3 Разработка клиентского интерфейса для работы с базой данных

Интерфейс для взаимодействия пользователей с базой данных склада запчастей должен соответствовать некоторым требованиям, а именно: быть интуитивно понятным и позволять каждому пользователю работать с базой в соответствии с его полномочиями. То есть, должны быть предусмотрены различные роли входа, для каждой роли должны быть реализованы свои функции. У владельца склада должен быть удобный способ добавления, изменения и удаления информации о сотрудниках склада, контрагентах и деталях, что хранятся на складе. Кладовщик должен иметь возможность изменения статуса накладной, а также изменения, добавления и удаления информации о деталях на складе. Менеджер склада должен иметь доступ к просмотру контрагентов, а также добавлению, изменению и удалению накладных.

3.1 Выбор среды для разработки приложения

Выбор языка программирования и среды разработки является важным шагом в создании приложения для работы с СУБД PostgreSQL. Основными критериями для выбора являются: поддержка необходимых библиотек для работы с базой данных, удобство разработки и отладки, производительность и кроссплатформенность.

Одним из популярных языков, предлагающих мощные и стабильные инструменты для работы с этой СУБД, является Python. Он известен своей простотой и лаконичностью, что позволяет разработчикам быстро писать и тестировать код. Этот язык также поддерживает широкий набор инструментов для тестирования и отладки, таких как встроенный модуль unittest и сторонние библиотеки pytest. Средства отладки в Python позволяют

Изм.	Лист	№ докум.	Подпись	Дата

эффективно отслеживать ошибки и находить узкие места в приложении, что критически важно при работе с базой данных.

Для работы с СУБД PostgreSQL Python предлагает множество библиотек. Например, библиотека psycopg2 предоставляет полный функционал для взаимодействия с PostgreSQL, включая транзакции, выполнение SQL-запросов и поддержку современных особенностей СУБД, таких как работа с JSONB или географическими данными (через PostGIS).

Руthon имеет достаточно высокий уровень производительности при работе с базами данных благодаря использованию эффективных драйверов и возможностей асинхронной обработки. Кроссплатформенность Руthon позволяет разворачивать приложения на различных операционных системах, что делает его идеальным выбором для проектов, рассчитанных на работу в разных окружениях (Linux, Windows, macOS).

Для обеспечения удобной разработки на Python рекомендуется использовать текстовый редактор Visual Studio Code (VS Code). Этот редактор обладает рядом преимуществ: интеграция с Python, плагины и расширяемость. Visual Studio Code также является кроссплатформенным инструментом, что обеспечивает возможность разработки и развертывания приложений на любой операционной системе без потери функциональности и удобства.

3.2 Описание основных модулей и функций интерфейса

Для реализации графического интерфейса и взаимодействия с СУБД PostgreSQL было разработано несколько ключевых классов и методов.

Класс WindowApp является базовым классом для работы с окнами приложения. Он содержит метод auth_window, который создает окно авторизации с полями для ввода логина и пароля, кнопкой входа и обработчиком событий. Метод create_connection устанавливает соединение с

Изм.	Лист	№ докум.	Подпись	Дата

базой данных PostgreSQL, используя переданные учетные данные. Метод start_work обрабатывает попытку входа пользователя, проверяет учетные данные и либо открывает главное окно приложения (WarehouseApp), либо выводит сообщение об ошибке.

Класс WarehouseApp наследует от WindowApp и реализует основную логику работы приложения. В его конструкторе инициализируется главное окно, устанавливается соединение с базой данных, определяются права пользователя и создаются вкладки интерфейса. Метод determine_user_permissions проверяет права текущего пользователя на доступ к различным разделам приложения.

Приложение поддерживает работу с несколькими сущностями через отдельные вкладки: накладные (create_invoice_tab), склад (create_warehouse_tab), контрагенты (create_counteragent_tab) и сотрудники (create_employee_tab). Для каждой сущности реализованы CRUD-операции (добавление, редактирование, удаление, поиск).

Особое внимание уделено системе отката изменений. Методы save_initial_state, undo_last_operation и rollback_to_initial_state позволяют сохранять состояние системы на момент входа пользователя и откатывать изменения, используя данные из таблицы логов. Для работы с резервными копиями реализованы методы create_backup и restore_from_backup.

Для работы со структурой склада (склады, комнаты, стеллажи, полки) используется метод edit_warehouse_structure, который открывает отдельное окно с вкладками для каждой сущности. Методы add_structure_item, edit_structure_item и delete_structure_item обеспечивают редактирование структуры с проверкой целостности данных.

Приложение поддерживает систему поиска с сохранением условий (current_search_conditions) и возможностью сброса. Для каждой сущности реализованы отдельные методы поиска (search_invoice, search_warehouse_item_u_t.д.).

Изм.	Лист	№ докум.	Подпись	Дата

Методы работы с накладными (add_invoice, edit_invoice, delete_invoice) включают проверку доступности деталей на складе и ограничение на количество активных накладных у сотрудника (не более 5). Аналогичные методы реализованы для работы с другими сущностями (контрагентами, сотрудниками, деталями на складе).

Приложение завершает работу через метод logout, который корректно закрывает соединение с базой данных. Статусная строка в нижней части окна отображает текущее состояние системы и информацию о загруженных данных.

UML-диаграмма классов представлена в приложении ВИсходный код приложения представлен в Приложении Г.

3.3 Пример работы графического интерфейса

После запуска программы на экране пользователя появляется окно для авторизации пользователя в системе, в котором необходимо ввести верные логин и пароль, иначе будет выдана ошибка, что показано на рисунках 10 и 11.

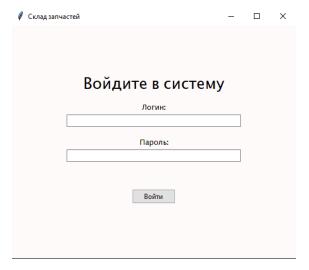


Рисунок 10 – Окно авторизации

Изм.	Лист	№ докум.	Подпись	Дата

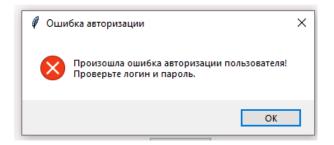


Рисунок 11 – Ошибка авторизации пользователя

Если пользователь ввёл верный логин и пароль, то на экране появиться главное окно приложения, что изображено на рисунке 12.

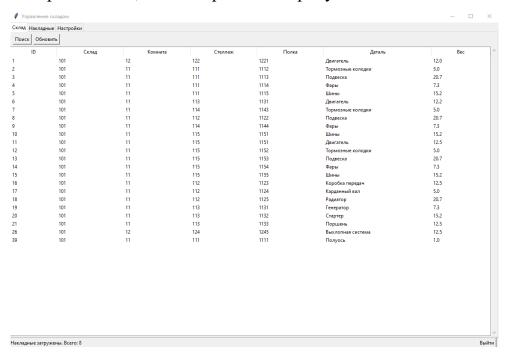


Рисунок 12 – Главное окно приложения

В верхней панели главного окна находятся вкладки таблиц базы данных, количество которых отличается от роли пользователя. Здесь же находятся кнопки «Поиск» и «Обновить». В нижней панели окна находится строка состояния приложения и кнопка «Выйти». В центральной части окна находится таблица с данными. Нажав на кнопку «Поиск», пользователь может заполнить от одного до нескольких полей для поиска необходимой записи в открытой таблице, что показано на рисунке 13 на примере поиска детали в таблице «Склад».

Изм.	Лист	№ докум.	Подпись	Дата

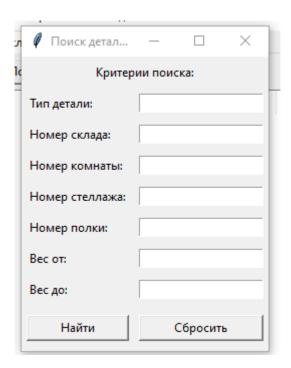


Рисунок 13 – Поиск детали в таблице «Склад»

Во вкладке «Настройки» для пользователей clerk (Кладовщик) и manager (Менеджер) находятся кнопки отката системы, что показаны на рисунке 14. Пользователь может откатить одну последнюю операцию или полностью откатить систему к началу текущего сеанса.

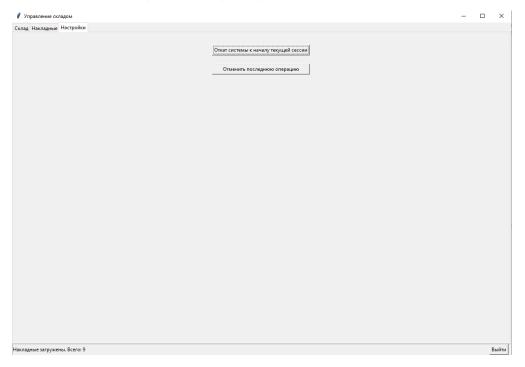


Рисунок 14 – Кнопки отката системы

Изм.	Лист	№ докум.	Подпись	Дата

Для пользователя owner (Владелец склада) находятся дополнительные кнопки резервного копирования, загрузки резервной копии и редактирования структуры склада, что показано на рисунке 15.

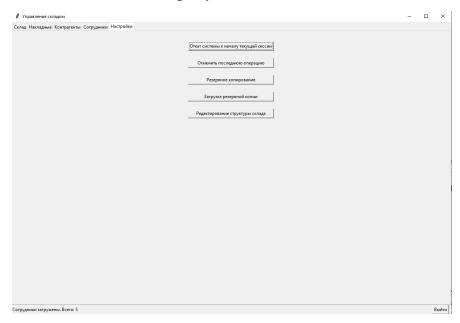


Рисунок 15 – Вкладка «Настройки» для пользователя owner

Нажав на кнопку «Редактирование структуры склада» главное окно приложения закроется и будет открыто новое окно, через которое пользователь сможет добавить, изменить или удалить существующие склады, комнаты, стеллажи и полки. Данное окно изображено на рисунке 16.

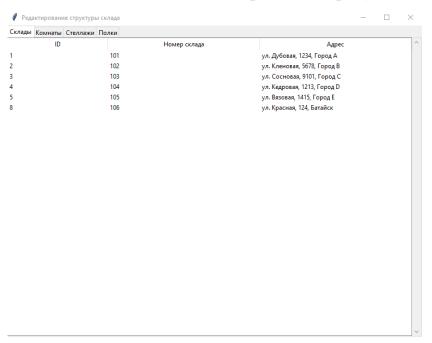


Рисунок 16 – Окно редактирования структуры склада

Изм.	Лист	№ докум.	Подпись	Дата

Нажав на крестик в правом верхнем углу окна на экране пользователя появиться главное окно приложение.

Нажав ПКМ по любой записи в любой таблице пользователю, будет представлен список возможных действий, что изображено на рисунке 17.

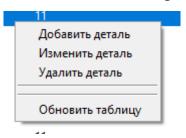


Рисунок 17 – Список доступных действий

Нажав на кнопку «Добавить» на экране пользователя появиться дополнительное окно с полями, каждое из которых обязательно к заполнению, что показано на рисунке 18.

Добавить детал	–		\times	
Номер склада:	101		~	
Номер комнаты:	11		~	
Номер стеллажа:	111		~	
Номер полки:	1111		~	
Тип детали:				
Bec (κr):	0.0			
Сохранить				

Рисунок 18 – Добавление новой записи

Если пользователь не заполнил все поля или неверно ввел данные, то на экране появиться ошибки, что изображены на рисунках 19 и 20 соответственно.

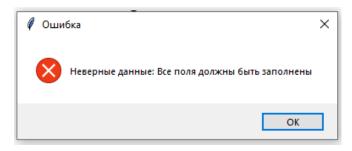


Рисунок 19 – Ошибка заполнения полей

Изм.	Лист	№ докум.	Подпись	Дата

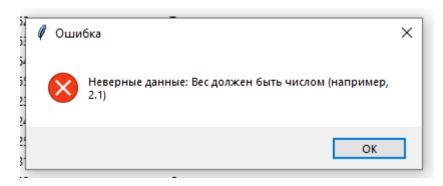


Рисунок 20 – Неверно введенные данные

Если пользователь ввел все данные правильно, то в таблице появиться новая запись.

Если пользователь нажмет на кнопку «Изменить», то на экране появиться дополнительное окно, в котором поля будут автоматически заполнены данными выделенной записи в таблице, что показано на рисунке 21.

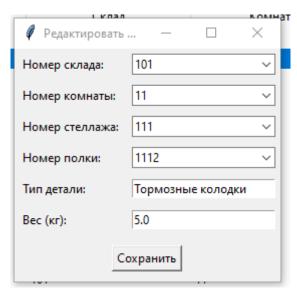


Рисунок 21 – Изменение записи в таблице

Если пользователь нажмет на кнопку «Удалить» или на клавишу Delete на клавиатуре, то на экране пользователя появиться уведомление, что изображено на рисунке 22.

Изм.	Лист	№ докум.	Подпись	Дата

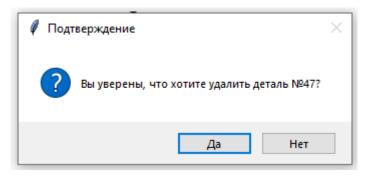


Рисунок 22 – Подтверждение удаления записи

Если пользователь нажмет «Нет», то запись не будет удалена. Если нажмет «Да», то на экране появиться уведомление об удачном или неудачном удалении записи, что показано на рисунках 23 и 24.

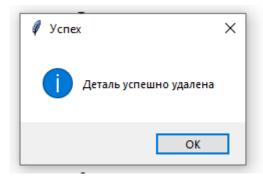


Рисунок 23 – Уведомление об успешном удалении записи

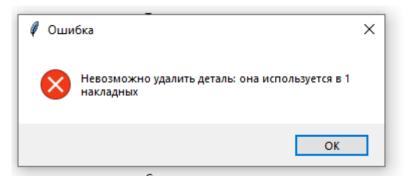


Рисунок 24 – Уведомление об ошибке удаления

Нажав на кнопку «Выйти» в нижней панели главного окна или на крестик в правом верхнем углу экрана, пользователя вернет в окно авторизации, а соединение с базой данных будет закрыто. Нажав на крестик в правом верхнем углу окна авторизации, пользователь закончит работу программы.

Исходный код программного средства представлен в Приложении В.

Изм.	Лист	№ докум.	Подпись	Дата

Заключение

В рамках данной курсовой работы была поставлена задача проектирования базы данных для склада запчастей и разработки приложения для работы с ней.

Для решения поставленной задачи была изучена предметная область, определены конечные пользователи, описаны объекты и их атрибуты. Также была разработана концептуальная модель в виде ER-диаграммы. Составлена даталогическая схема отношений. В ходе выполнения данной работы была изучена выбранная СУБД PostgreSQL, ее возможности и функционал.

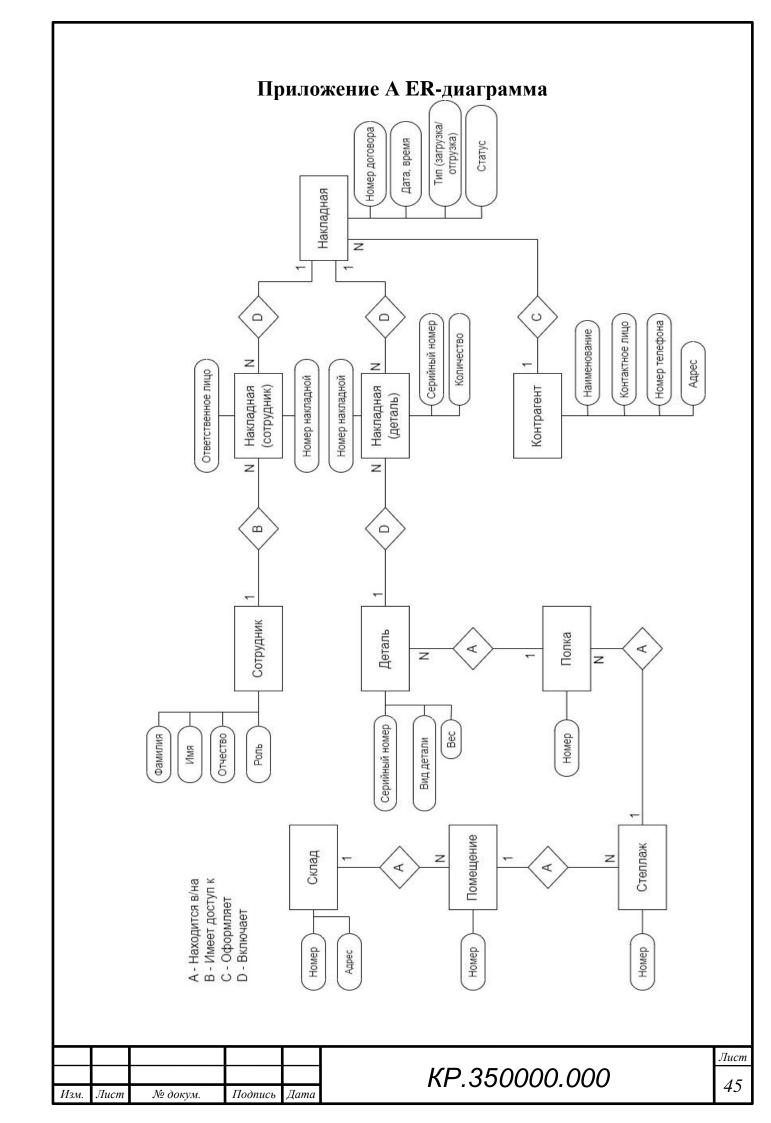
В результате была спроектирована и реализована база данных склада запчастей и реализован набор интерфейсов для работы с ней. Так же были проведены необходимые тестирование и отладка разработанного программного обеспечения.

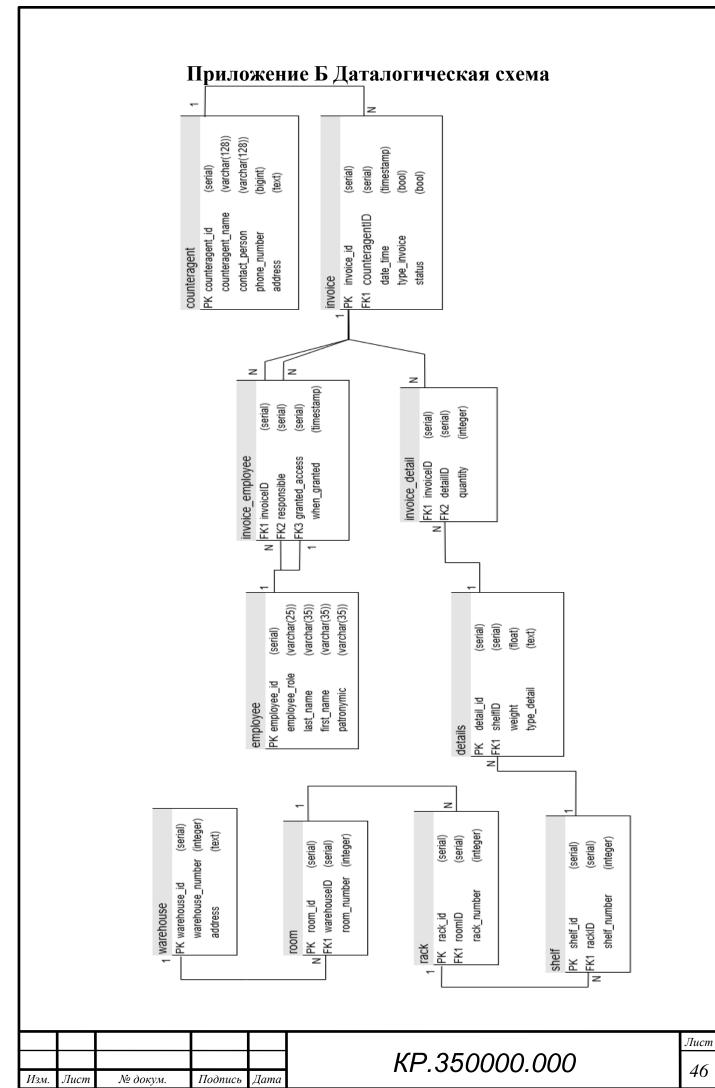
Изм.	Лист	№ докум.	Подпись	Дата

Перечень используемых информационных источников

- 1. Дейт, Кристофер Введение в системы баз данных, 8-е издание / Пер. с англ. М.: Издательский дом «Вильяме», 2005 1328 с.: ил. Парал. тит. англ. ISBN 5-8459-0788-8
- 2. Паттон Джефф. Пользовательские истории. Искусство гибкой разработки ПО. СПб.: Питер, 2019. 288 с.
- 3. Карпова Т.С. Базы данных: модели, разработка, реализация. СПб.: Питер, 2001. 304 с.
- 4. Конноли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика, 2-е изд.: Пер. с англ. М.: Издательский дом «Вильямс», 2000. 1120 с.
- 5. Глушаков С.В., Ломотько Д.В. Базы данных: Учебный курс: OOO «Издательство ACT», 2000 504 с. ISBN 996-03-0992-9

Изм.	Лист	№ докум.	Подпись	Дата





Приложение В UML-диаграмма классов программного средства

WindowApp

app: NoneType

auth_window()
create_connection(login, password)
create_context_menu(event, tree, menu_items)
create_settings_tab()
start_work(window, name, password)

WarehouseApp can_access_counteragents : bool can_access_details: bool can_access_employees: bool can_access_invoices: bool can_access_warehouse : bool can_edit_counteragents : bool can_edit_details: bool can_edit_employees: bool can_edit_invoices : bool can_edit_warehouse : bool can_update_invoice_status : bool counteragent tree: Treeview current_search_conditions: dict current_user cursor employee_tree : Treeview initial_state : NoneType initial_state_time : NoneType invoice tree: Treeview is_search_active : bool logout_btn : Button notebook : Notebook sort_states : dict status_bar : Label user_password warehouse_tree : Treeview

add_counteragent() add_employee() add invoice() add_structure_item(table_name, columns_en, columns_ru, callback) add_warehouse_item() create_backup() create_counteragent_tab() create employee tab() create_invoice_tab() create_structure_tab(parent, table_name, columns_ru) create_warehouse_tab() delete_counteragent() delete_employee() delete_structure_item(tree, table_name, callback)
delete_warehouse_item() determine_user_permissions()
edit_counteragent() edit_employee() edit_invoice() edit_structure_item(tree, table_name, columns_en, columns_ru, callback) edit_warehouse_item()
edit_warehouse_structure() get_primary_key(table_name) get table order() load_available_details(combobox) load_counteragents() load_employees() load_invoices() load_warehouse() logout() on_close()
perform_detail_search(listbox, search_text) refresh_affected_tab(table_name)
refresh_all_tabs() reset_and_refresh(tree, sort_col, condition_key) reset_sort(tree, sort_col)
restore_from_backup() rollback_to_initial_state()
save initial_state() search_counteragent() search_detail(detail_var) search_employee() search_invoice() search warehouse item() select_detail(listbox, detail_var, window) sort treeview(tree, col, reverse, initial order col)

undo_last_operation() update_invoice_status()

	·			
Изм.	Лист	№ докум.	Подпись	Дата

Приложение Г Исходный код программного средства

Листинг 1 – Исходный код программного средства

```
import psycopg2
                                          y=middle window y+30,
from tkinter import *
                                          anchor="center")
from tkinter import ttk, messagebox,
                                                  entry password
                                          ttk.Entry(width=50, show="*")
filedialog
from datetime import datetime
                                          entry password.place(x=middle window
import os
import subprocess
                                                         y=middle window y+90,
                                          х,
                                         anchor="center")
class WindowApp:
                                                 btn in
                                         ttk.Button (text="Войти",
   def __init__(self):
        self.app = None
                                         command=lambda:
                                         self.start work(window, entry_name,
    def auth window(self):
                                         entry password))
        window = Tk()
                                         btn in.place(x=middle window x,
window.geometry('%dx%d+%d+%d'
                                         y=middle window y+160,
(500, 400,
                                         anchor="center")
                                                 window.mainloop()
(window.winfo_screenwidth()/2)
(500/2),
                                             def
                                                      create_connection(self,
                                         login, password):
(window.winfo_screenheight()/2)
                                                      connection
        window.title("Склад
                                         psycopg2.connect(
запчастей")
                                                         host="127.0.0.1",
                                                         user=login,
window.configure(background="#FFFAFA
                                                         password=password,
                                         database="Warehouse DB"
       middle window x = 500 / 2
       middle_window_y = 400 / 3
        title start
                                                     print("[INFO] PostgreSQL
ttk.Label(master=window,
                                         connection open.")
text="Войдите в систему",
                                                     return connection
                                                 except Exception as ex:
font=("algerian",
                               20),
                                                     print(f"[CONNECTION
background="#FFFAFA")
                                          ERROR] Failed to connect: {ex}")
                                                     return None
title start.place(x=middle window x,
y=100, anchor="center")
                                             def start work(self, window,
       title login
                                         name, password):
ttk.Label(text="Логин:",
                                                             password
                                                 login,
font=("algerian",
                               10),
                                         name.get(), password.get()
background="#FFFAFA")
                                                 print(f"[AUTH] Attempting
                                          login for user: {login}")
                                                 active_user
title_login.place(x=middle_window_x,
y=140, anchor="center")
                                         self.create_connection(login,
       title_password
                                         password)
ttk.Label(text="Пароль:",
                                                 if active_user is not None:
                               10),
font=("algerian",
                                                     print("[AUTH]
background="#FFFAFA")
                                         successful")
                                                     window.destroy()
                                                     main win = Tk()
title password.place(x=middle window
_x, y=200, anchor="center")
                                                     self.app
       entry name
                                         WarehouseApp (main win,
                                                                      login,
ttk.Entry(width=50)
                                         password, active_user)
                                                     main win.mainloop()
entry name.place(x=middle window x,
                                                 else:
```

Изм.	Лист	№ докум.	Подпись	Дата

```
print("[AUTH] Login
                                         te({'employee':
                                                                      None }),
failed")
                                         self.app.load employees()])
messagebox.showerror('Ошибка
                                                 trv:
авторизации',
                                         menu.tk popup(event.x root,
            ошибка авторизации
'Произошла
                                         event.y_root)
пользователя! Проверьте логин и
                                                 finally:
пароль.')
                                                     menu.grab release()
         create context menu(self,
                                             def create settings tab(self):
   def
                                                 """Создает вкладку настроек
event, tree, menu items):
       """Создание
                                         для пользователя"""
                     KOHTEKCTHOFO
меню для Treeview с добавлением
                                                 tab = Frame(self.notebook)
кнопки обновления"""
                                                 self.notebook.add(tab,
       item
                                         text="Hастройки")
tree.identify row(event.y)
       if not item:
                                                 frame = Frame(tab)
           return
                                                 frame.pack(pady=20)
       tree.selection set(item)
                                                 btn rollback = Button(frame,
       menu = Menu(self.root,
                                         text="Откат системы к началу текущей
tearoff=0)
                                         сессии",
        for
             label, command in
menu items:
                                         command=self.rollback to initial sta
           if command:
                                                 btn rollback.pack(pady=10,
                                         fill=X)
menu.add command(label=label,
command=command)
           else:
                                                 btn undo = Button(frame,
               menu.add separator()
                                         text="Отменить последнюю операцию",
                                         command=self.undo last operation)
       menu.add separator()
        if
                                                 btn undo.pack(pady=10,
                   tree
self.invoice_tree:
                                         fill=X
menu.add_command(label="Обновить
                                                 if
                                                    self.current user
                                         'owner':
таблицу",
              command=lambda:
[self.current search conditions.upda
                                                    btn backup
te({'invoice':
                                                        text="Резервное
                           None}),
                                         Button (frame,
self.app.load invoices()])
                                         копирование",
       elif
                   tree
self.warehouse tree:
                                         command=self.create backup)
                                                    btn backup.pack(pady=10,
menu.add command(label="Обновить
                                         fill=X)
таблицу",
                   command=lambda:
[self.current\_search\_conditions.upda
                                                    btn restore
te({'warehouse':
                                         Button(frame,
                                                          text="Загрузка
                           None}),
self.app.load_warehouse()])
                                         резервной копии",
       elif
                   tree
self.counteragent_tree:
                                         command=self.restore from backup)
menu.add command(label="Обновить
                                         btn restore.pack(pady=10, fill=X)
таблицу",
                   command=lambda:
[self.current_search_conditions.upda
                                                     btn_edit_warehouse
                                         Button(frame, text="Редактирование
te({'counteragent':
                         None }),
self.app.load counteragents()])
                                         структуры склада",
       elif
                    tree
self.employee tree:
                                         command=self.edit warehouse structur
menu.add_command(label="Обновить
таблицу",
                    command=lambda:
                                         btn edit warehouse.pack(pady=10,
[self.current_search_conditions.upda
                                         fill=X)
```

Изм.	Лист	№ докум.	Подпись	Дата

```
class WarehouseApp(WindowApp):
                                         self.determine user permissions()
def __init__(self, root, login,
password, active_user):
       self.root = root
                                          self.save initial state()
        self.root.title("Управление
складом")
                                                     self.status bar
                                         Label (root, text=f"Вход выполнен
self.root.geometry("1200x800")
                                         как: {login} | Готово", bd=1,
        self.current user = login
                                         relief=SUNKEN, anchor=W)
        self.user password
                                          self.status bar.pack(side=BOTTOM,
password
        self.initial state = None
                                          fill=X)
        self.is search active
False
                                                      self.logout btn
        self.sort states = {}
                                         Button(self.status bar,
                                         text="Выйти", command=self.logout)
self.current search conditions = {
                                         self.logout btn.pack(side=RIGHT,
            'invoice': None,
                                         padx=5)
            'warehouse': None,
            'counteragent': None,
                                                      self.notebook
            'employee': None
                                         ttk.Notebook(root)
        }
                                          self.notebook.pack(fill=BOTH,
                                          expand=True)
self.root.protocol("WM DELETE WINDOW
", self.on close)
                                                      if
                                         self.can access warehouse:
        try:
            if active user:
                                          self.create warehouse tab()
               self.conn
active user
                                          self.can_access_invoices:
            else:
               try:
                    self.conn
                                          self.create_invoice_tab()
psycopg2.connect(
host="127.0.0.1",
                                          self.can access counteragents:
                       user=login,
                                          self.create counteragent tab()
password=password,
database="Warehouse DB"
                                          self.can_access_employees:
                   print("[INFO]
                                         self.create employee tab()
PostgreSQL connection open.")
               except Exception as
                                          self.create settings tab()
                    print(f"[ERROR]
Connection failed: {ex}")
                                                  except psycopg2.Error as e:
                                                     print(f"[TRANSACTION
messagebox.showerror("Ошибка
                                         ERROR] Initialization failed: {e}")
подключения", f"He удалось
             к базе данных:
подключиться
                                          messagebox.showerror("Ошибка
{str(ex)}")
                                          подключения",
                                                         f"He удалось
                                                         к базе
                                          подключиться
                                                                      данных:
self.root.destroy()
                                          {str(e)}")
                   return
                                                     self.root.destroy()
            self.cursor
                                              def logout(self):
                                                  """Выход из системы
self.conn.cursor()
                                          возврат к окну авторизации"""
```

```
messagebox.askyesno("Подтверждение",
                                         command=lambda:
"Вы уверены, что хотите выйти из
                                         self.perform detail search(detail li
системы?"):
                                         stbox, search_var.get()))
            if hasattr(self, 'conn')
                                                 search btn.pack(side=LEFT)
and self.conn:
                self.conn.close()
                                                 detail listbox
                print("[INFO]
                                         Listbox(search window)
PostgreSQL connection closed.")
                                          detail listbox.pack(fill=BOTH,
                                         expand=True, padx=5, pady=5)
            self.root.destroy()
                                                 select btn
            self.auth window()
                                         Button (search window,
                                         text="Выбрать",
    def on close(self):
       """Обработчик закрытия окна
- выполняет выход из системы"""
                                         command=lambda:
       self.logout()
                                          self.select detail (detail listbox,
                                          detail var, search window))
   def load available details (self,
                                                 select btn.pack(pady=5)
combobox):
        """Загружает
                             список
доступных деталей в комбобокс"""
                                         self.perform detail search(detail li
                                          stbox, "")
           self.cursor.execute("""
               SELECT DISTINCT
                                         search entry.bind("<Return>", lambda
type detail
               FROM details
               ORDER BY type detail
                                         self.perform detail search(detail li
                                         stbox, search var.get()))
            details = [row[0] for
row in self.cursor.fetchall()]
                                         detail listbox.bind("<Double-Button-
           combobox['values']
                                                         lambda
                                         self.select detail(detail listbox,
details
        except Exception as e:
                                         detail_var, search_window))
           print(f"[ERROR] Failed
to load details: {e}")
                                             def perform_detail_search(self,
                                          listbox, search_text):
                                                  """Выполняет поиск деталей
                search detail(self,
   def
                                          по заданному тексту"""
detail_var):
       """Открывает
                      окно поиска
                                                 listbox.delete(0, END)
деталей на складе"""
                                                 try:
                                                     query = """
       search window
Toplevel(self.root)
                                                         SELECT
                                                                    DISTINCT
       search window.title("Выбор
                                        type detail
                                                         FROM details
                                                         WHERE type detail
search window.geometry("400x300")
                                         ILIKE %s
                                                         ORDER BY type detail
       search frame
Frame(search window)
       search_frame.pack(fill=X,
                                         self.cursor.execute(query,
padx=5, pady=\overline{5})
                                          (f"%{search text}%",))
        search_var = StringVar()
                                                     for
                                                                row
                                                                           in
        search entry
                                         self.cursor.fetchall():
Entry(search frame,
                                                         listbox.insert(END,
textvariable=search var)
                                         row[0])
        search entry.pack(side=LEFT,
                                                 except Exception as e:
expand=True, fill=X, padx=(0, 5))
                                                    print(f"[ERROR] Failed
                                         to search details: {e}")
        search btn
Button(search frame, text="Найти",
```

```
def select detail(self, listbox,
                                                items
detail_var, window):
                                        list(tree.get children(''))
       """Выбирает деталь из
                                               try:
списка"""
                                                   items.sort(key=lambda x:
       selection
                                        int(tree.set(x, sort col)))
listbox.curselection()
                                                except ValueError:
       if not selection:
                                                    items.sort(key=lambda x:
                                        tree.set(x, sort col).lower())
messagebox.showwarning("Предупрежден
ие", "Выберите деталь из списка")
                                                for index, item
                                                                        in
           return
                                        enumerate(items):
                                                   tree.move(item,
       selected detail
                                       index)
listbox.get(selection[0])
                                               for col in tree['columns']:
detail var.set(selected detail)
                                                   tree.heading(col,
       window.destroy()
                                        command=lambda
                                                                     c=col:
                                        self.sort treeview(tree, c, False,
   def sort_treeview(self, tree,
                                        sort col))
col,
                          reverse,
initial order col=None):
                                                   reset and refresh(self,
       tree, sort col, condition key):
столбцу"""
                                               """Сброс сортировки
                                        обновление данных"""
       data = [(tree.set(item,
       item) for item in
tree.get children('')]
                                        self.current search conditions.updat
                                        e({condition key: None})
       trv:
          data.sort(key=lambda x:
                                                self.reset sort(tree,
float(x[0]), reverse=reverse)
                                        sort col)
       except ValueError:
          data.sort(key=lambda x:
                                               if
                                                          tree
                                        self.invoice tree:
x[0].lower(), reverse=reverse)
                                                 self.load invoices()
       for index, (val, item) in
                                                elif
                                                            tree
                                        self.warehouse tree:
enumerate(data):
                                                  self.load warehouse()
           tree.move(item,
index)
                                                elif
                                                      tree
                                        self.counteragent tree:
       tree.heading(col,
                                        self.load counteragents()
command=lambda:
self.sort treeview(tree, col, not
                                               elif
                                        self.employee tree:
reverse, initial_order_col))
                                                   self.load employees()
       if initial order col:
                                           def get table order(self):
                                               """Возвращает порядок таблиц
tree.heading(initial order col,
                                        для отката с учетом зависимостей"""
command=lambda:
                                               return [
self.reset sort(tree, col))
                                                    'invoice detail',
                                                    'invoice employee',
                                                    'invoice',
       tree.heading(col, text=col +
(' ↓' if reverse else ' ↑'))
                                                    'details',
                                                    'counteragent',
   def
         reset sort(self, tree,
                                                    'employee',
                                                    'shelf',
sort_col):
       """Сброс сортировки
                                                    'rack',
                                                    'room',
первоначальному состоянию"""
       for col in tree['columns']:
                                                    'warehouse',
          tree.heading(col,
                                                    'log_table'
text=col)
                                                1
```

Изм.	Лист	№ докум.	Подпись	Дата

Лист

```
def
         refresh affected tab(self,
                                          "He
                                                         определить
                                              удалось
                                                                      начало
                                          сессии")
table name):
       """Обновляет
                                                         return
                          вклалку.
                         измененной
соответствующую
таблице"""
                                                      self.cursor.execute("""
        if
                                                          SELECT
                                                                 record id
                                not.
self.is_search_active:
                                          FROM log table
           if table_name
                                                         WHERE action type =
['invoice',
                  'invoice_detail',
                                          'UNDO'
'invoice employee']
                                                         AND action time >=
hasattr(self, 'invoice tree'):
                                          용S
               self.load invoices()
                                                          ORDER BY log id DESC
            elif table name ==
                                                         LIMIT 1
'details'
                     hasattr(self,
              and
                                          (self.initial_state_time,))
'warehouse tree'):
                                                      last undo
self.load warehouse()
                                          self.cursor.fetchone()
           elif table name
                                                      last undo id
'counteragent'
               and hasattr(self,
                                          last undo[0] if last undo else None
'counteragent tree'):
                                                      query = """
self.load counteragents()
                                                          SELECT
                                                                  lt.log id,
                                          lt.table_name, lt.action_type,
                  table name
                                                              lt.record id,
'employee'
                      hasattr(self,
                                          lt.old values, lt.new values
'employee tree'):
                                                         FROM log_table lt
                                                         WHERE lt.action_type
self.load employees()
       else:
                                          NOT IN ('UNDO', 'ROLLBACK')
                                                         AND lt.action time
           pass
                                          >= %s
                                                      11 11 11
    def refresh all tabs(self):
        """Обновляет все вкладки
                                                      params
приложения"""
                                          [self.initial state time]
        i f
                                not.
                                                      if last undo id:
self.is_search_active:
            if
                                                          query +=
                     hasattr(self,
                                                                           AND
                                          lt.log id > %s"
'invoice_tree'):
                self.load invoices()
            if
                      hasattr(self,
                                         params.append(last undo id)
'warehouse_tree'):
                                                      query += """
self.load warehouse()
                                                          ORDER BY
            if
                       hasattr(self,
                                                              CASE table name
'counteragent_tree'):
                                                                 WHEN
                                          'invoice_detail' THEN 1
self.load counteragents()
                      hasattr(self,
                                          'invoice employee' THEN 2
            if
'employee tree'):
                                                                  WHEN
                                          'invoice' THEN 3
self.load employees()
                                                                  WHEN
       else:
                                          'details' THEN 4
           pass
                                                                  WHEN
                                          'counteragent' THEN 5
    def undo_last_operation(self):
                                                                  WHEN
        """Отменяет
                                          'employee' THEN 6
                    последнюю
операцию с учетом зависимостей"""
                                                                  WHEN 'shelf'
                                          THEN 7
       try:
            if
               not
                     hasattr(self,
                                                                  WHEN 'rack'
'initial state time')
                         or not
                                          THEN 8
self.initial state time:
                                                                  WHEN 'room'
                                          THEN 9
messagebox.showinfo("Информация",
                                                                  WHEN
                                          'warehouse' THEN 10
```

```
ELSE 11
                                                                       WHERE
                                           invoice id = %s
                    END,
                                                                   11 11 11
                    CASE action_type
                        WHEN
                                           (invoice id,))
'DELETE' THEN 1
                                                                   i f
                                                                            not
                                          self.cursor.fetchone():
                        WHEN
'UPDATE' THEN 2
                        WHEN
                                          messagebox.showwarning("Предупрежден
'INSERT' THEN 3
                        ELSE 4
                    END,
                                          "Невозможно
                                                          отменить
                                                                      операцию:
                    log_id DESC
                                          связанная накладная уже удалена")
                LIMIT 1
            ** ** **
                                          self.conn.rollback()
                                                                       return
self.cursor.execute(query, params)
                                          self.cursor.execute(f"""
            last op
self.cursor.fetchone()
                                                                   DELETE FROM
                                          {table}
            if not last op:
                                                                   WHERE \{pk\} =
messagebox.showinfo("Информация",
                                                                   RETURNING *
                                                               """,
"Нет операций для отмены в текущей
сессии")
                                           (record id,))
                return
                                                           elif action
            log id, table, action,
                                           'DELETE' and old values:
record id, old values, new values =
                                                                    table
                                                                             in
last op
                                           ['invoice detail',
                                           'invoice employee'] and 'invoiceid'
                                          in old values:
            рk
self.get primary key(table)
            if not pk:
                                          self.cursor.execute("""
                                                                       SELECT 1
messagebox.showerror("Ошибка",
                                f"He
                                          FROM invoice
удалось определить первичный
                                                                       WHERE
                                ключ
для таблицы {table}")
                                           invoice id = %s
                return
                                           (old values['invoiceid'],))
                                                                   i f
                                                                            not
self.cursor.execute("BEGIN")
                                          self.cursor.fetchone():
            try:
                                          messagebox.showwarning("Предупрежден
                if
                     action
                                          ие",
'INSERT':
                    if table in
                                          "Невозможно
                                                         отменить
                                                                      операцию:
['invoice detail',
                                          связанная накладная не существует")
'invoice employee']:
                                           self.conn.rollback()
self.cursor.execute(f"""
                                                                       return
                            SELECT
invoiceid FROM {table}
                                                               columns
                                           '.join(old_values.keys())
                            WHERE
\{pk\} = %s
                                                               values
                        """,
                                           '.join(['%s'] * len(old values))
(record id,))
                        invoice id =
                                          self.cursor.execute(f"""
self.cursor.fetchone()[0]
                                                                   INSERT INTO
                                           {table} ({columns})
self.cursor.execute("""
                                                                   VALUES
                            SELECT 1
                                           ({values})
FROM invoice
                                                                   RETURNING *
```

Изм.

Лист

№ докум.

Подпись Дата

```
""",
                                                  """Сохраняет информацию
list(old values.values()))
                                         начальной точке отката"""
                                                 try:
                elif
                      action
'UPDATE' and old values:
                                         self.cursor.execute("SELECT
                    set_clause = ',
                                         current timestamp")
'.join([f"{k}
              = %s" for k in
                                                      self.initial state time
old values.keys()])
                                          = self.cursor.fetchone()[0]
                                                     print(f"[INFO]
                                                                         Saved
self.cursor.execute(f"""
                                         initial
                                                          state
                                                                         time:
                        UPDATE
                                          {self.initial state time}")
                                                  except Exception as e:
{table}
                        SET
                                                     print(f"[ERROR] Failed
                                          to save initial state: {e}")
{set clause}
                        WHERE \{pk\} =
                                                     self.initial state time
응S
                                          = None
                       RETURNING *
                    11 11 11
                                             def
list(old values.values())
                                         rollback to initial state(self):
[record id])
                                                  """Откатывает изменения
                                                      зависимостей
                                         vчетом
                                                                        между
                                          таблицами"""
                import json
               old values json
                                                                           not
                                         self.initial_state_time:
json.dumps(new values) if new values
else None
               new values json
                                         messagebox.showwarning("Предупрежден
json.dumps(old values) if old values
                                         ие", "Не удалось
                                                                   определить
                                         начальное состояние сессии")
                                                     return
self.cursor.execute("""
                    INSERT
                               INTO
                                         messagebox.askyesno("Подтверждение",
log table
                    (table name,
                                         уверены, что хотите откатить все
action type, record id, old values,
                                         изменения текущей сессии?"):
new values)
                                                     trv:
                   VALUES
                                (%s,
                                          self.cursor.execute("""
'UNDO', %s, %s, %s)
                """, (table, log_id,
                                                              SELECT
old values json, new values json))
                                          table name, action type, record id,
                                          old values
                self.conn.commit()
                                                              FROM log table
                                                             WHERE
self.refresh_affected_tab(table)
                                         action time >= %s
                                                             AND log id
messagebox.showinfo("Усπех",
                                         COALESCE ((SELECT MAX(log id) FROM
"Последняя
                                          log table WHERE
              операция
                            успешно
                                                             action type
отменена")
                                          'ROLLBACK'), 0)
                                                             ORDER BY
            except Exception as e:
                                                                 CASE
                self.conn.rollback()
                                          table name
                                                                     WHEN
messagebox.showerror("Ошибка", f"He
                                          'invoice detail' THEN 1
удалось
           отменить операцию:
                                                                     WHEN
{str(e)}")
                                          'invoice employee' THEN 2
                                                                     WHEN
        except Exception as e:
                                          'invoice' THEN 3
                                                                      WHEN
messagebox.showerror("Ошибка",
                                          'details' THEN 4
f"Ошибка при отмене операции:
                                                                     WHEN
{str(e)}")
                                          'counteragent' THEN 5
                                                                     WHEN
    def save initial state(self):
                                          'employee' THEN 6
```

KP.350000.000

№ докум.

Подпись

Дата

Изм.

Лист

Лист

```
WHEN
'shelf' THEN 7
                                           WHERE \{pk\} = %s
                                                                            """,
                            WHEN
'rack' THEN 8
                                           (record id,))
                            WHEN
'room' THEN 9
                                                                   elif action
                            WHEN
                                           == 'DELETE' and old values:
'warehouse' THEN 10
                                                                       if table
                            ELSE 11
                                           == 'invoice detail' or table ==
                        END,
                                           'invoice employee':
                        CASE
                                                                            if
                                           'invoiceid' in old values:
action type
                            WHEN
'DELETE' THEN 1
                                           self.cursor.execute("""
                            WHEN
'UPDATE' THEN 2
                                           SELECT 1 FROM invoice
                            WHEN
'INSERT' THEN 3
                                           WHERE invoice id = %s
                            ELSE 4
                        END,
                                           """, (old values['invoiceid'],))
                        log id DESC
                """,
                                           if not self.cursor.fetchone():
(self.initial state time,))
                                          print(f"[WARNING] Пропуск вставки в
                changes
                                          {table}, invoice не существует")
self.cursor.fetchall()
                                          continue
                if not changes:
messagebox.showinfo("Информация",
                                          self.cursor.execute(f"""
"Нет изменений для отката")
                    return
                                          SELECT 1 FROM {table}
                                          WHERE \{pk\} = %s
self.cursor.execute("BEGIN")
                                           (record id,))
                for table, action,
                                                                       if not.
record id, old values in changes:
                                          self.cursor.fetchone():
                        рk
                                           columns
                                           '.join(old values.keys())
self.get primary key(table)
                        if not pk:
                                           values
                                                             '.join(['%s']
                                           len(old values))
print(f"[WARNING]
                     Не
                             удалось
определить РК для таблицы {table}")
                            continue
                                          self.cursor.execute(f"""
                        if action ==
                                          INSERT INTO {table} ({columns})
'INSERT':
                                           VALUES ({values})
                                                                            """,
self.cursor.execute(f"""
                                           list(old_values.values()))
SELECT 1 FROM {table}
                                                                   elif action
WHERE \{pk\} = %s
                                           == 'UPDATE' and old values:
                             """,
(record id,))
                                           set clause = ', '.join([f"\{k\} = %s"
                            if
                                          for k in old values.keys()])
self.cursor.fetchone():
                                          self.cursor.execute(f"""
self.cursor.execute(f"""
                                          UPDATE {table}
DELETE FROM {table}
```

№ докум. Изм. Лист Подпись Дата

```
SET
                                                      'invoice detail':
                                          'invoiceID',
{set clause}
                                                      'employee':
WHERE \{pk\} = %s
                                          'employee_id',
                            """,
                                                      'invoice employee':
                                          'invoiceID',
list(old values.values())
[record id])
                                                      'log table': 'log id'
                    except
psycopg2.Error as e:
                                                  if table name in pk mapping:
                                                      return
print(f"[WARNING] Failed to revert
                                          pk_mapping[table name]
{action} on {table}.{record id}:
{e}")
                                                  try:
                                                      self.cursor.execute("""
self.conn.rollback()
                                                         SELECT a.attname
                                                          FROM pg index i
self.cursor.execute("SAVEPOINT
                                                          JOIN pg_attribute a
                                          ON a.attrelid = i.indrelid AND
rollback continue")
                        continue
                                          a.attnum = ANY(i.indkey)
                                                         WHERE i.indrelid =
                                          %s::regclass AND i.indisprimary
                                                     """, (table_name,))
self.cursor.execute("""
                    INSERT
                               INTO
                                                     result
log table (table name, action type,
                                         self.cursor.fetchone()
record id)
                                                     return result[0]
                                                                            if
                   VALUES
                                         result else None
('SYSTEM', 'ROLLBACK', %s)
                                                 except:
                                                     return None
(len(changes),))
                                              def create backup(self):
                                                  """Создает резервную копию
               self.conn.commit()
                                          базы данных"""
                                                  try:
self.refresh all tabs()
                                                      backup file
                                          filedialog.asksaveasfilename(
                                          defaultextension=".backup",
messagebox.showinfo("Успех",
                                                         filetypes=[("Backup
"Система успешно откачена к началу
                                          files", "*.backup"), ("All files",
сессии")
                                          "*.*")],
            except Exception as e:
                                                         title="Сохранить
                                          резервную копию как"
messagebox.showerror("Ошибка", f"He
удалось выполнить откат: {str(e)}")
               self.conn.rollback()
                                                      if not backup file:
                                                          return
              get primary key(self,
                                                      env = os.environ.copy()
table name):
       """Возвращает имя первичного
                                                      env['PGPASSWORD']
                                          '12345'
ключа
      для
           таблицы с обработкой
исключений"""
       pk_mapping = {
                                                      command = [
            'warehouse':
                                                         r'C:\Program
'warehouse id',
                                          Files\PostgreSQL\17\bin\pg_dump',
            'room': 'room id',
                                                          '-h', '127.0.0.1',
                                                          '-U', 'postgres',
            'rack': 'rack_id',
            'shelf': 'shelf id',
                                                          '-d',
            'details': 'detail id',
                                         'Warehouse DB',
                                                          '-F', 'C',
            'counteragent':
                                                          '-f', backup file
'counteragent id',
            'invoice': 'invoice_id',
```

Изм.	Лист	№ докум.	Подпись	Дато

```
process
                                                         r'C:\Program
subprocess. Popen (command,
                                         Files\PostgreSQL\17\bin\pg restore',
                                                         '-h', '127.0.0.1', '-U', 'postgres',
env=env.
                                                         '-d',
stdout=subprocess.PIPE,
                                         'Warehouse_DB',
                                                         '-c',
stderr=subprocess.PIPE)
                                                         backup file
           stdout, stderr
process.communicate()
                                                     process
                                         subprocess. Popen (command,
           if process.returncode ==
0:
                                         env=env,
messagebox.showinfo("Успех",
                                         stdout=subprocess.PIPE,
f"Резервная копия успешно создана:
{backup file}")
                                         stderr=subprocess.PIPE)
                                                     stdout,
                                                               stderr
               error msg
stderr.decode('utf-8') if stderr
                                        process.communicate()
else "Неизвестная ошибка"
                                                     if process.returncode ==
messagebox.showerror("Ошибка", f"He
                                         0:
удалось создать резервную копию:
                                         messagebox.showinfo("Успех", "База
{error msg}")
       except Exception as e:
                                         данных успешно восстановлена из
                                         резервной копии")
messagebox.showerror("Ошибка",
f"Ошибка при создании резервной
                                                        self.root.destroy()
копии: {str(e)}")
                                                        main win = Tk()
   def restore from backup(self):
                                         WarehouseApp (main win,
       """Восстанавливает базу
                                         self.current user,
данных из резервной копии"""
                                         self.user password, None)
                                                        main_win.mainloop()
       try:
           backup file
                                                     else:
filedialog.askopenfilename(
                                                        error msg
               filetypes=[("Backup
                                        stderr.decode('utf-8') if stderr
files", "*.backup"), ("All files",
                                         else "Неизвестная ошибка"
"*.*")],
               title="Выберите файл
                                         messagebox.showerror("Ошибка", f"He
резервной копии"
                                         удалось восстановить базу данных:
                                         {error msg}")
                                                        self.conn
           if not backup file:
                                         psycopg2.connect(
               return
                                        host="127.0.0.1",
           if
messagebox.askyesno("Подтверждение",
                                        user=self.current user,
"Вы уверены, что хотите восстановить
                                        password=self.user password,
базу данных из резервной копии? Все
текущие данные будут потеряны!"):
                                         database="Warehouse DB"
               return
                                                        self.cursor
           self.conn.close()
                                         self.conn.cursor()
                                                except Exception as e:
           env = os.environ.copy()
           env['PGPASSWORD'] =
                                         messagebox.showerror("Ошибка",
'12345'
                                         f"Ошибка при восстановлении
                                                                          из
                                         резервной копии: {str(e)}")
           command = [
                                                     try:
```

```
self.conn
                                                 scroll
psycopg2.connect(
                                         ttk.Scrollbar(parent,
                                         command=tree.yview)
host="127.0.0.1",
                                                 scroll.pack(side=RIGHT,
                                         fill=Y)
user=self.current user,
                                         tree.configure(yscrollcommand=scroll
password=self.user password,
database="Warehouse DB"
                                                 tree.pack(fill=BOTH,
                                         expand=True)
               self.cursor
self.conn.cursor()
                                                 def
                                         load_data(reset_search=False):
           except:
                                                     """Загружает данные с
               self.root.destroy()
                                         учетом текущих условий поиска или
                                         сбрасывает их"""
   def create structure tab(self,
parent, table_name, columns_ru):
                                                     if reset search:
       """Создает
                    вкладку для
редактирования структуры склада с
                                         self.current search conditions[table
русскими названиями"""
                                         _name] = None
       columns map = {
           "warehouse": {
               "Номер
                          склада":
                                        tree.delete(*tree.get children())
"warehouse number",
               "Адрес": "address"
                                         self.current_search_conditions.get(t
            "room": {
                                         able name):
               "Склад":
                                                         conditions
"warehouseid",
                                         self.current search conditions[table
               "Номер
                         комнаты":
                                         name]['conditions']
"room number"
                                                         params
                                         self.current search conditions[table
            "rack": {
                                         name]['params']
                                                         query = f"SELECT *
               "Комната": "roomid",
                                         FROM {table_name} WHERE " + " AND
               "Номер
                        стеллажа":
                                         ".join(conditions) + " ORDER BY 1"
"rack_number"
            "shelf": {
                                         self.cursor.execute(query, params)
               "Стеллаж": "rackid",
                                                     else:
               "Номер
                           полки":
"shelf number"
                                         self.cursor.execute(f"SELECT * FROM
                                         {table name} ORDER BY 1")
                                                     for
       columns en
                                         self.cursor.fetchall():
[columns map[table name][col]
                                                         formatted row
                                for
col in columns ru]
                                         [row[0]]
                                                              i, col
       tree = ttk.Treeview(parent,
                                         enumerate(columns en):
columns=("id",
                     *columns ru),
show="headings")
                                         col.endswith("id"):
       tree.heading("id",
                                                                ref table =
text="ID")
                                         col.replace("id", "")
       for col in columns ru:
                                         self.cursor.execute(f"SELECT
           tree.heading(col,
                                         {ref table} number FROM {ref table}
text=col)
                                         WHERE {ref table} id
                                                                   = %s",
                                         (row[i+1],))
           tree.column(col,
width=150
                                                                ref number =
                                        self.cursor.fetchone()
       tree.column("id", width=50)
```

Изм.	Лист	№ докум.	Подпись	Дата

```
if
formatted row.append(ref number[0] search text:
if ref number else "N/A")
                                                                     if
                                        col.endswith("id"):
                   else:
formatted_row.append(row[i+1])
                                        ref_table = col.replace("id", "")
               tree.insert("", END,
                                         self.cursor.execute(f"""
values=formatted row)
        def search items():
                                         SELECT
                                                   {ref table} id
                                                                        FROM
           search window
                                         {ref table}
Toplevel(self.root)
                                         WHERE {ref table} number = %s
search window.title(f"Поиск
{table name}")
                                         (search text,))
           Label (search window,
                                         ref id = self.cursor.fetchone()
text="Критерии поиска:").grid(row=0,
                                                                         if
column=0, columnspan=2, pady=5)
                                         ref id:
                                         conditions.append(f"{col} = %s")
           search vars = []
           for i, col
enumerate(columns ru):
                                         params.append(ref id[0])
               Label(search window,
                                                                     else:
text=f"{col}:").grid(row=i+1,
column=0, padx=5, pady=5, sticky=W)
                                         conditions.append(f"{col}::text LIKE
               var = StringVar()
                                         %s")
                                         params.append(f"%{search text}%")
columns_en[i].endswith("id"):
                   ref table
columns_en[i].replace("id", "")
                                         self.current search conditions[table
                                         name] = {
self.cursor.execute(f"SELECT
{ref_table}_number FROM {ref_table}
                                         'conditions': conditions,
ORDER BY {ref_table}_number")
                                                                 'params':
                   options
                                         params
[str(row[0])
                for
                       row
                                 in
self.cursor.fetchall()]
                                                             load data(False)
                   combo
ttk.Combobox(search window,
textvariable=var, values=options)
                                         search window.destroy()
combo.grid(row=i+1,
                         column=1,
                                                         except Exception as
padx=5, pady=5, sticky=EW)
               else:
                                         messagebox.showerror("Ошибка",
Entry(search window,
                                         f"Ошибка поиска: {str(e)}")
textvariable=var).grid(row=i+1,
column=1, padx=5, pady=5, sticky=EW)
                                        self.conn.rollback()
                                                    Button(search window,
                                         text="Найти",
search vars.append(var)
                                         command=perform search).grid(
           def perform search():
                                         row=len(columns_ru)+1,
                                                                  column=0,
               trv:
                   conditions = []
                                        padx=5, pady=10, sticky=EW)
                                                    Button(search window,
                   params = []
                                         text="Сбросить", command=lambda:
                   for i, col in
                                         [self.current_search_conditions.upda
                                         te({table name:
enumerate(columns en):
                                                                    None }),
                                         load data(False),
                       search text
= search vars[i].get()
                                         search_window.destroy()]).grid(
```

№ докум. Изм. Лист Подпись Дата

```
row=len(columns ru)+1, column=1, edit window.geometry("800x600")
padx=5, pady=10, sticky=EW)
                                                self.root.withdraw()
       menu = Menu(parent,
tearoff=0)
                                                def on edit window close():
                                                    self.root.deiconify()
menu.add command(label="Добавить",
                                                    edit window.destroy()
command=lambda:
self.add structure item(table name,
columns en, columns ru, lambda:
                                        edit window.protocol("WM DELETE WIND
load data(False)))
                                        OW", on edit window close)
menu.add command(label="Изменить",
                                                notebook
                                                                           =
command=lambda:
                                        ttk.Notebook(edit window)
self.edit_structure item(tree,
                                                notebook.pack(fill=BOTH,
table name, columns en, columns ru,
                                        expand=True)
lambda: load data(False)))
                                                warehouse tab
menu.add command(label="Удалить",
                                      Frame(notebook)
command=lambda:
                                                notebook.add(warehouse tab,
self.delete_structure item(tree,
                                        text="Склады")
table name,
                         lambda:
load data(False)))
                                        self.create structure tab(warehouse
                                        tab, "warehouse", ["Номер склада",
       menu.add separator()
                                         "Адрес"])
menu.add command(label="Найти",
command=search items)
                                                room tab = Frame(notebook)
       menu.add separator()
                                                notebook.add(room tab,
                                        text="Комнаты")
menu.add command(label="Обновить
список",
                   command=lambda:
                                         self.create structure tab(room tab,
                                         "room", ["Склад", "Номер комнаты"])
load data(False))
menu.add_command(label="Обновить
                                                rack tab = Frame(notebook)
таблицу",
                 command=lambda:
                                                notebook.add(rack tab,
                                         text="Стеллажи")
load data(True))
                                         self.create structure tab(rack tab,
       def
                                         "rack", ["Komhata", "Homep
show context menu(event):
                                        стеллажа"])
           item
tree.identify row(event.y)
           if item:
                                                 shelf tab = Frame(notebook)
                                                notebook.add(shelf tab,
tree.selection set(item)
                                        text="Полки")
menu.post(event.x root,
                                         self.create structure tab(shelf tab,
                                         "shelf", ["Стеллаж", "Номер полки"])
event.y_root)
       tree.bind("<Button-3>",
                                                   add structure item(self,
                                         table name, columns en, columns ru,
show context menu)
       load data(False)
                                         callback):
                                                """Добавляет новый элемент
   def
                                         структуры склада с проверкой
                                         уникальности"""
edit warehouse structure(self):
       """Редактирование структуры
                                               add window
склада (склады, комнаты, стеллажи,
                                       Toplevel(self.root)
полки) """
                                                add window.title(f"Добавить
       edit window
                                        в {table name}")
Toplevel(self.root)
                                                entries = []
edit window.title("Редактирование
                                                labels = []
структуры склада")
```

```
for i, (col en, col ru) in
                                                              if
enumerate(zip(columns en,
                                          self.cursor.fetchone():
columns ru)):
                                                                  raise
                                          ValueError(f"Склад
                                                                 ПΟ
                                                                        адресу
labels.append(Label(add window,
                                          {address} уже существует")
text=col ru))
            labels[-1].grid(row=i,
                                                          elif table name
                                          "room":
column=0, padx=5, pady=5, sticky=W)
                                                              room number
                                          entries[1][0].get()
            if
col_en.endswith("id"):
                                                              warehouse id
                                          entries[0][0].get().split("(ID:
                ref table
col en.replace("id", "")
                                          ")[1].replace(")", "").strip()
self.cursor.execute(f"SELECT
                                          self.cursor.execute("""
{ref_table}_id, {ref_table}_number
                                                                  SELECT
                                                                              1
FROM {ref_table}")
                                          FROM room
                options = [f"{num}
                                                                  WHERE
     {id})"
               for id,
                                          room number = %s AND warehouseID =
self.cursor.fetchall()]
                                                              """,
                var = StringVar()
                                          (room number, warehouse id))
ttk.Combobox(add window,
                                          self.cursor.fetchone():
textvariable=var, values=options)
                                                                  raise
                                          ValueError(f"Комната
                combo.grid(row=i,
                                                                  С
                                                                       номером
column=1, padx=5, pady=5, sticky=EW)
                                          {room number} уже существует в этом
                entries.append((var,
                                          складе")
True, col en))
                                                          elif table name
            else:
                                          "rack":
                entrv
                                                              rack number
Entry(add window)
                entry.grid(row=i,
                                          entries[1][0].get()
column=1, padx=5, pady=5, sticky=EW)
                                                              room id
                                          entries[0][0].get().split("(ID:
                                          ")[1].replace(")", "").strip()
entries.append((entry,
                             False,
col_en))
                                          self.cursor.execute("""
                                                                  SELECT
        def save item():
                                          FROM rack
            try:
                values = []
                parent id = None
                                          rack number = %s AND roomID = %s
                    table name
                                          (rack number, room id))
"warehouse":
                    warehouse number
                                          self.cursor.fetchone():
= entries[0][0].get()
                                                                  raise
                                          ValueError(f"Стеллаж
                                                                  С
                                                                      номером
self.cursor.execute("SELECT 1 FROM
                                          {rack number} уже существует в этой
warehouse WHERE warehouse number =
                                          комнате")
%s", (warehouse number,))
                    address
                                                          elif table name
                                          "shelf":
entries[1][0].get()
                                                              shelf number
self.cursor.execute("SELECT 1
                                FROM
                                          entries[1][0].get()
warehouse WHERE address =
                                %s",
                                                              rack id
(address,))
                                          entries[0][0].get().split("(ID:
                                          ")[1].replace(")", "").strip()
self.cursor.fetchone():
                       raise
                                          self.cursor.execute("""
ValueError(f"Склад
                                                                  SELECT
                                                                              1
                      С
                         номером
{warehouse number} уже существует")
                                          FROM shelf
```

 Изм.
 Лист
 № докум.
 Подпись
 Дата

```
if not selected:
                       WHERE
shelf number = %s AND rackID = %s
                   """,
                                         messagebox.showwarning("Предупрежден
                                         ие", "Выберите
(shelf number, rack id))
                                                            запись для
                   if
                                         редактирования")
self.cursor.fetchone():
                                                    return
                       raise
ValueError(f"Полка
                     с номером
                                                 item
{shelf number} уже существует на
                                         tree.item(selected[0])
этом стеллаже")
                                                 item id = item['values'][0]
                                                 edit window
               for entry, is combo,
                                         Toplevel(self.root)
col en in entries:
                                                edit window.title(f"Изменить
                   if is combo:
                                         запись в {table name}")
                       val
entry.get().split("(ID:
")[1].replace(")", "").strip()
                                                self.cursor.execute(f"SELECT
                                             FROM {table name} WHERE
                                         {table name}_id = %s", (item_id,))
values.append(val)
                                                current data
                   else:
                                         self.cursor.fetchone()
values.append(entry.get())
                                                entries = []
               columns str =
                                                labels = []
".join(columns en)
               placeholders = ",
                                                for i, (col en, col_ru) in
".join(["%s"] * len(columns_en))
                                         enumerate(zip(columns_en,
                                         columns ru)):
               self.cursor.execute(
                   f"INSERT INTO
                                         labels.append(Label(edit window,
{table_name} ({columns_str}) VALUES
                                         text=col ru))
                                                    labels[-1].grid(row=i,
({placeholders})",
                                         column=0, padx=5, pady=5, sticky=W)
                   values
                                         col_en.endswith("id"):
               self.conn.commit()
                                                        ref table
               add window.destroy()
                                         col en.replace("id", "")
messagebox.showinfo("Успех", "Запись
                                         self.cursor.execute(f"SELECT
успешно добавлена")
                                         {ref table} id, {ref table} number
           except ValueError as ve:
                                         FROM {ref_table}")
                                                        options = [f"{num}
messagebox.showerror("Ошибка",
                                         (ID: {id})"
str(ve))
                                                        for id, num in
           except Exception as e:
                                         self.cursor.fetchall()]
               self.conn.rollback()
messagebox.showerror("Ошибка", f"He
                                         self.cursor.execute(f"SELECT
удалось добавить запись: {str(e)}")
                                         {ref_table}_number FROM {ref_table}
                                         WHERE {ref_table}_id
       Button (add window,
                                         (current data[i+1],))
text="Сохранить",
                                                        current_ref
command=save item).grid(
                                         self.cursor.fetchone()
           row=len(columns_en),
                                                        current value
column=0, columnspan=2, pady=10)
                                         f"{current ref[0]}
                                                                         (ID:
                                         {current data[i+1]})" if current ref
   def
         edit structure item(self,
                                         else ""
        table_name, columns_en,
tree,
columns_ru, callback):
                                                        var
       """Редактирует
                                         StringVar(value=current value)
                           элемент
структуры склада без автоматического
                                                         combo
обновления"""
                                         ttk.Combobox(edit window,
       selected = tree.selection()
                                         textvariable=var, values=options)
```

Изм.

Лист

№ докум.

Подпись

Дата

```
combo.grid(row=i,
column=1, padx=5, pady=5, sticky=EW)
                                          room number = %s AND warehouseID =
                                          %s AND room_id != %s
                entries.append((var,
True, col en))
           else:
                                          (room number,
                                                                 warehouse id,
                                          item_id))
                entry
Entry(edit window)
                                                              i f
                entry.insert(0,
                                         self.cursor.fetchone():
str(current data[i+1]))
                                                                  raise
                                          ValueError(f"Комната
                entry.grid(row=i,
                                                                 с номером
                                          {room number} уже существует в этом
column=1, padx=5, pady=5, sticky=EW)
                                          складе")
entries.append((entry,
                             False,
col en))
                                                          elif table name
                                          "rack":
        def save_changes():
                                                              rack number
           try:
                                          entries[1][0].get()
                if
                     table name
                                                              room id
"warehouse":
                                          entries[0][0].get().split("(ID:
                                          ")[1].replace(")", "").strip()
                    new address
entries[1][0].get()
                                          self.cursor.execute("""
self.cursor.execute("""
                                                                  SELECT
                        SELECT
                                         FROM rack
FROM warehouse
                                                                  WHERE
                                          rack number = %s AND roomID = %s AND
                       WHERE
address = %s AND warehouse id != %s
                                          rack id != %s
                                                              """,
                                          (rack number, room id, item id))
(new address, item id))
                    warehouse number
= entries[0][0].get()
                                          self.cursor.fetchone():
                                                                  raise
self.cursor.execute("""
                                          ValueError(f"Стеллаж
                                                                 С
                                                                      номером
                        SELECT
                                          {rack number} уже существует в этой
                                          комнате")
FROM warehouse
                        WHERE
                                                          elif table name
warehouse number
                         %S
                                 AND
                                          "shelf":
warehouse id != %s
                    """
                                                              shelf number
(warehouse number, item id))
                                          entries[1][0].get()
                                                              rack id
self.cursor.fetchone():
                                          entries[0][0].get().split("(ID:
                                          ")[1].replace(")", "").strip()
                       raise
ValueError(f"Склад
                      ПО
                             адресу
                                          self.cursor.execute("""
{new_address} yme cymectbyet")
                                                                  SELECT
self.cursor.fetchone():
                                          FROM shelf
                       raise
ValueError(f"Склад
                                          shelf number = %s AND rackID = %s
                     с номером
                                          AND shelf id != %s
{warehouse number} уже существует")
                                                              """
                elif table name ==
                                          (shelf number, rack id, item id))
"room":
                    room number
                                          self.cursor.fetchone():
entries[1][0].get()
                                                                  raise
                   warehouse id
                                          ValueError(f"Полка
                                                                С
                                                                      номером
entries[0][0].get().split("(ID:
                                          {shelf number} уже существует на
")[1].replace(")", "").strip()
                                          этом стеллаже")
self.cursor.execute("""
                                                          values = []
                       SELECT
                                                          for entry, is_combo,
FROM room
                                          col en in entries:
                                                              if is combo:
```

```
SELECT 1 FROM
entry.get().split("(ID:
                                         details WHERE shelfID IN (
")[1].replace(")", "").strip()
                                                                 SELECT
                                         shelf id FROM shelf WHERE rackID IN
values.append(val)
                   else.
                                                                     SELECT
                                         rack id FROM rack WHERE roomID IN (
values.append(entry.get())
                                         SELECT room id FROM room WHERE
                set_clause = ",
                                         warehouseID = %s
".join([f"\{col\} = %s" for col in
                                                                     )
                                                                 )
columns en])
                                                             ) LIMIT 1
                                                         """, (item_id,))
                self.cursor.execute(
                   f"UPDATE
                                                     elif table name
                                         "room":
{table name} SET {set clause} WHERE
{table_name}_id = %s",
                                         self.cursor.execute("""
                   values
[item id]
                                         details WHERE shelfID IN (
                self.conn.commit()
                                         shelf id FROM shelf WHERE rackID IN
edit window.destroy()
                                                                     SELECT
                                         rack id FROM rack WHERE roomID = %s
messagebox.showinfo("Успех", "Запись
успешно обновлена")
                                                            ) LIMIT 1
                                                         """, (item_id,))
           except ValueError as ve:
                                                     elif
                                                            table name
messagebox.showerror("Ошибка",
                                         "rack":
str(ve))
           except Exception as e:
                                         self.cursor.execute("""
               self.conn.rollback()
                                                             SELECT 1
                                         details WHERE shelfID IN (
messagebox.showerror("Ошибка", f"He
                                                                 SELECT
                                         shelf id FROM shelf WHERE rackID =
удалось обновить запись: {str(e)}")
                                                         ) LIMIT 1 """, (item_id,))
       Button (edit window,
text="Сохранить",
                                                     elif
command=save changes).grid(
                                                            table name
                                         "shelf":
           row=len(columns en),
column=0, columnspan=2, pady=10)
                                         self.cursor.execute("""
   def delete_structure_item(self,
                                                             SELECT 1 FROM
tree, table name, callback):
                                         details WHERE shelfID = %s LIMIT 1
       """Удаляет элемент структуры
                                                         """, (item id,))
склада с обработкой ошибок"""
       selected = tree.selection()
                                                     has details
       if not selected:
                                         self.cursor.fetchone()
messagebox.showwarning("Предупрежден
                                                     if has_details:
ие", "Выберите запись для удаления")
                                                         if
                                                                          not
                                         messagebox.askyesno(
           return
                                                             "Подтверждение",
                                                             "Это действие
       item
tree.item(selected[0])
                                         удалит все
                                                         связанные
                                                                      детали.
       item id = item['values'][0]
                                         Продолжить?"
                                                         ) :
                                                             return
        trv:
           if
                  table name
"warehouse":
                                                    self.conn.commit()
self.cursor.execute("""
```

```
messagebox.showinfo("Успех", "Запись
                                         self.can access invoices = True
успешно удалена")
                                                             if privilege in
                                         ('INSERT', 'UPDATE', 'DELETE'):
        except psycopg2.Error as e:
           self.conn.rollback()
           error_msg = f"Ошибка
                                         self.can edit invoices = True
удаления: {str(e)}"
           if
                                                         if table
                "нет доступа"
                                                                        in
                                in
                                                         'room', 'rack',
str(e):
                                          ('warehouse',
                                 +=
                                         'shelf', 'warehouse_details_view'):
               error msg
"\n\nНедостаточно
                     прав
                                для
                                         self.can_access_warehouse = True
выполнения операции. Обратитесь к
администратору."
                                                             if privilege in
                                         ('INSERT', 'UPDATE', 'DELETE'):
messagebox.showerror("Ошибка",
error msg)
                                         self.can edit warehouse = True
   def
                                                         if
                                                               table
                                                                           ==
                                         'counteragent':
determine user permissions(self):
       self.can access invoices
                                         self.can access counteragents = True
False
       self.can access warehouse =
                                                             if privilege in
                                         ('INSERT', 'UPDATE', 'DELETE'):
False
self.can access counteragents
                                         self.can edit counteragents = True
False
        self.can access employees
                                                         if
                                                                table
                                         'employee':
False
       self.can access details
False
                                         self.can access employees = True
                                                             if privilege in
                                         ('INSERT', 'UPDATE', 'DELETE'):
       self.can edit invoices
False
       self.can edit warehouse
                                         self.can edit employees = True
False
                                                         if table
       self.can edit counteragents
                                         'details':
= False
       self.can edit employees
                                         self.can access details = True
False
                                                             if privilege in
        self.can edit details
                                         ('INSERT', 'UPDATE', 'DELETE'):
False
self.can update invoice status =
                                         self.can edit details = True
False
                                                     if self.current user ==
       try:
                                         'clerk':
            self.cursor.execute("""
               SELECT table name,
                                         self.can edit invoices = False
privilege type
                                         self.can update invoice status
                FROM
information schema.table privileges
                                         True
               WHERE grantee = %s
                                                 except psycopg2.Error as e:
('warehouse '+self.current user,))
                                                     print(f"[PERMISSION
                                         ERROR] Failed to check permissions:
           permissions
                                         {e}")
self.cursor.fetchall()
                                         self.status bar.config(text=f"Ошибка
           for table, privilege in
                                         проверки прав доступа: {str(e)}")
                                                except psycopg2.Error as e:
permissions:
                if
                                                     print(f"[VIEW
                                                                      ERROR]
                       table in
('invoice', 'invoice details view'):
                                         Failed to access view: {e}")
```

```
messagebox.showerror("Ошибка", "Не
                                         self.invoice tree.column("Тип",
удалось получить доступ к данным.
                                         width=100)
Проверьте права доступа.")
                                         self.invoice tree.column("CTaTyc",
    def create invoice tab(self):
                                         width=100)
        """Создание вкладки для
работы с накладными с учетом прав
                                         self.invoice tree.column("Деталь",
доступа"""
                                         width=150)
       tab = Frame(self.notebook)
        self.notebook.add(tab,
                                         self.invoice tree.column("Кол-во",
text="Накладные")
                                         width=70)
       button_frame = Frame(tab)
button_frame.pack(fill=X,
                                         self.invoice tree.column("Ответствен
                                         ный", width=150)
padx=5, pady=5)
                                                 scroll = ttk.Scrollbar(tab,
       search btn
                                         command=self.invoice tree.yview)
Button (button_frame, text="Поиск",
                                                 scroll.pack(side=RIGHT,
command=self.search invoice)
                                         fill=Y)
       search btn.pack(side=LEFT,
padx=2)
                                         self.invoice tree.configure(yscrollc
                                         ommand=scroll.set)
       refresh btn
Button (button frame,
text="Oбновить",
                                         self.invoice tree.pack(fill=BOTH,
                   command=lambda:
                                         expand=True)
self.reset and refresh(self.invoice
tree, "ID", 'invoice'))
                                                 menu items = []
                                                 if self.can edit invoices:
       refresh btn.pack(side=LEFT,
padx=2)
                                                     menu items.extend([
                                                          ("ID",
                                         накладную", self.add invoice),
       columns
                             "Тип",
                "Дата",
                                                         ("Изменить
"Контрагент",
"Статус", "Деталь",
                          "Кол-во",
                                         накладную", self.edit invoice),
                                                         ("Удалить
"Ответственный")
                                         накладную", self.delete invoice),
       self.invoice tree
                                                          (None, None)
ttk.Treeview(tab, columns=columns,
show="headings")
                                                 elif
        for col in columns:
                                          self.can update invoice status:
                                                     menu items.extend([
                                                          ("Обновить статус",
self.invoice tree.heading(col,
text=col,
                                         self.update invoice status),
                                                         (None, None)
command=lambda
                             c=col:
self.sort treeview(
self.invoice tree, c, False, "ID"))
                                         self.invoice tree.bind("<Button-3>",
self.invoice tree.column(col,
                                         self.create context menu(e,
width=100)
                                         self.invoice tree, menu items))
                                         self.invoice tree.bind("<Delete>",
                                         lambda e: self.delete invoice())
self.invoice tree.column("ID",
width=50)
                                                 self.load invoices()
self.invoice tree.column("Контрагент
", width=150)
                                             def update invoice status(self):
                                                 selected
self.invoice_tree.column("Дата", self.invoice_tree.selection()
width=120)
                                                 if not selected:
```

Изм.	Лист	№ докум.	Подпись	Дата

Лист

```
messagebox.showwarning("Предупрежден
                                         SELECT
                                                        COUNT(*)
                                                                        FROM
ие", "Выберите
                                         invoice employee
                  накладную для
обновления статуса")
           return
                                         WHERE responsible = %s
                                                                         AND
                                         invoiceid IN (SELECT invoice id FROM
       try:
                                         invoice WHERE status = FALSE)
           item
self.invoice tree.item(selected[0])
                                                                         AND
           invoice id
                                         invoiceid != %s
item['values'][0]
                                         (employee id, invoice id))
           current status
item['values'][4] == 'Завершено'
                                         active invoices count
                                         self.cursor.fetchone()[0]
           status window
Toplevel(self.root)
status window.title("Обновление
                                         active invoices count >= 5:
статуса накладной")
                                         raise ValueError(
           Label(status window,
text="Новый статус:").pack(padx=5,
                                         f"Y
                                                      сотрудника
                                                                          уже
                                         {active invoices count}
pady=5)
                                                                     активных
                                         накладных. "
           status var = StringVar()
                                         "Максимум - 5. Нельзя изменить
status_var.set("Завершено"
                                         статус."
current status else "В процессе")
           status combobox
ttk.Combobox(status window,
                                         self.cursor.execute("""
textvariable=status var,
                                                                UPDATE
                                         invoice SET status = %s WHERE
values=["В процессе", "Завершено"])
                                         invoice id = %s
status combobox.pack(padx=5, pady=5)
                                         (new status, invoice id))
           def save_status():
                                         self.conn.commit()
                try:
                   new status
status_var.get() == "Завершено"
                                         self.load invoices()
                   if
                                not
                                         status window.destroy()
new status:
                                         messagebox.showinfo("Успех", "Статус
self.cursor.execute("""
                                         накладной обновлен")
                                                         except ValueError as
responsible FROM invoice employee
invoiceid = %s
                                         messagebox.showerror("Ошибка",
                       """,
                                         str(ve))
(invoice id,))
                                                         except Exception as
                       employee id
                                         e:
                                                            print(f"[UPDATE
= self.cursor.fetchone()
                                         ERROR] Failed to update invoice
                                         status: {e}")
employee id:
                                         self.conn.rollback()
employee id = employee id[0]
                                         messagebox.showerror("Ошибка", f"He
self.cursor.execute("""
                                         удалось обновить статус: {str(e)}")
```

				·
Изм.	Лист	№ докум.	Подпись	Дата

Лист

```
Button(status window,
text="Сохранить",
                                         self.warehouse tree.column("Склад",
command=save status).pack(pady=10)
                                         width=100)
       except Exception as e:
                                         self.warehouse tree.column("Комната"
           print(f"[STATUS UPDATE
                                        , width=100)
ERROR] Initial error: {e}")
                                         self.warehouse tree.column("Стеллаж"
                                         , width=100)
messagebox.showerror("Ошибка",
f"Ошибка при обновлении статуса:
{str(e)}")
                                         self.warehouse tree.column("Полка",
                                         width=100)
   def create warehouse tab(self):
       """Создание вкладки для
                                         self.warehouse tree.column("Деталь",
работы со складом с учетом прав
                                        width=200)
доступа"""
       tab = Frame(self.notebook)
                                        self.warehouse tree.column("Bec",
       self.notebook.add(tab,
                                         width=80)
text="Склад")
                                                scroll = ttk.Scrollbar(tab,
       button frame = Frame(tab)
                                        command=self.warehouse tree.yview)
       button frame.pack(fill=X,
                                                scroll.pack(side=RIGHT,
padx=5, pady=5)
                                         fill=Y)
                                         self.warehouse tree.configure(yscrol
       search btn
Button (button frame, text="Поиск",
                                         lcommand=scroll.set)
command=self.search warehouse item)
       search btn.pack(side=LEFT,
padx=2)
                                         self.warehouse tree.pack(fill=BOTH,
                                         expand=True)
       refresh btn
Button (button frame,
                                                 menu items = []
                                                 if self.can edit_warehouse:
text="Обновить",
                                                    menu items.extend([
                   command=lambda:
self.reset and refresh(self.warehous
                                                         e tree, "ID", 'warehouse'))
                                         self.add warehouse item),
                                                         ("\overline{\text{Изменить}} деталь",
       refresh btn.pack(side=LEFT,
                                         self.edit warehouse item),
padx=2)
                                                         columns = ("ID", "Склад",
                                         self.delete_warehouse item),
"Комната", "Стеллаж", "Деталь", "Вес")
                          "Полка",
                                                         (None, None)
                                                     ])
        self.warehouse tree
ttk.Treeview(tab, columns=columns,
show="headings")
                                         self.warehouse tree.bind("<Button-</pre>
                                                        lambda
        for col in columns:
                                         self.create context menu(e,
                                         self.warehouse tree, menu items))
self.warehouse tree.heading(col,
text=col,
                                         self.warehouse tree.bind("<Delete>",
                                         lambda
                                                                           e:
command=lambda
                            c=col:
                                         self.delete warehouse item())
self.sort treeview(
                                                 self.load warehouse()
self.warehouse tree, c, False,
"ID"))
                                             def
                                         create_counteragent_tab(self):
self.warehouse tree.column(col,
                                                 """Создание вкладки
                                                                         пля
                                         работы с контрагентами"""
width=100)
                                                tab = Frame(self.notebook)
                                                 self.notebook.add(tab,
self.warehouse_tree.column("ID",
                                        text="Контрагенты")
width=50)
```

```
button frame = Frame(tab)
       button frame.pack(fill=X,
                                        self.counteragent tree.column("Конта
padx=5, pady=5)
                                         кт", width=150)
       search btn
                                        self.counteragent tree.column("Телеф
Button (button frame, text="Поиск",
                                         он", width=120)
command=self.search counteragent)
        search btn.pack(side=LEFT,
                                         self.counteragent tree.column("Адрес
padx=2)
                                         ", width=250)
       refresh btn
                                                 scroll = ttk.Scrollbar(tab,
Button (button frame,
                                         command=self.counteragent tree.yview
text="Обновить",
                   command=lambda:
                                                 scroll.pack(side=RIGHT,
self.reset and refresh(self.countera
                                         fill=Y)
gent tree, "ID", 'counteragent'))
       refresh btn.pack(side=LEFT,
                                         self.counteragent tree.configure(ysc
padx=2)
                                         rollcommand=scroll.set)
       columns = ("ID", "Название",
"Контакт", "Телефон", "Адрес")
                                         self.counteragent tree.pack(fill=BOT
       self.counteragent tree
                                         H, expand=True)
ttk.Treeview(tab, columns=("ID",
"Название", "Контакт", "Телефон",
                                                 menu items = []
"Адрес"), show="headings")
                                         self.can edit counteragents:
       for col in columns:
                                                     menu items.extend([
                                                         ("Добавить
self.counteragent tree.heading(col,
                                        контрагента",
                                         self.add counteragent),
text=col,
                                                         ("Изменить
command=lambda
                                         контрагента",
                             c=col:
self.sort treeview(
                                         self.edit counteragent),
                                                         ("Удалить
                                         контрагента",
self.counteragent_tree, c, False,
"ID"))
                                         self.delete_counteragent),
                                                         (None, None)
self.counteragent tree.column(col,
                                                     ])
width=100)
                                         self.counteragent tree.bind("<Button</pre>
                                         -3>",
self.counteragent tree.heading("ID",
                                                         lambda
text="ID")
                                         self.create context menu(e,
                                         self.counteragent tree, menu items))
self.counteragent tree.heading("Hase
ание", text="Название")
                                         self.counteragent tree.bind("<Delete</pre>
                                                         lambda
self.counteragent tree.heading("Конт
                                         self.delete counteragent())
акт", text="Контактное лицо")
                                                 self.load counteragents()
self.counteragent tree.heading("Теле
фон", text="Телефон")
                                             def create employee tab(self):
                                                 """Создание вкладки для
self.counteragent_tree.heading("Адре работы с сотрудниками"""
c", text="Адрес")
                                                 tab = Frame(self.notebook)
                                                 self.notebook.add(tab,
                                         text="Сотрудники")
self.counteragent_tree.column("ID",
                                                 button_frame = Frame(tab)
width=50)
                                                 button_frame.pack(fill=X,
self.counteragent_tree.column("HasBa padx=5, pady=5)
ние", width=200)
```

Изм.	Лист	№ докум.	Подпись	Дата

Лист

```
search btn
Button (button frame, text="Поиск",
                                         self.employee tree.column("Имя",
command=self.search employee)
                                         width=120)
        search btn.pack(side=LEFT,
                                         self.employee tree.column("Отчество"
padx=2)
                                          , width=120)
        refresh btn
Button (button frame,
                                                  scroll = ttk.Scrollbar(tab,
text="Обновить",
                                         command=self.employee_tree.yview)
                    command=lambda:
                                                 scroll.pack(side=RIGHT,
self.reset and refresh(self.employee
                                         fill=Y)
tree, "ID", 'employee'))
       refresh btn.pack(side=LEFT,
                                         self.employee tree.configure(yscroll
padx=2)
                                         command=scroll.set)
       columns = ("ID", "Роль",
"Фамилия", "Имя", "Отчество")
                                          self.employee tree.pack(fill=BOTH,
       self.employee_tree
                                          expand=True)
ttk.Treeview(tab, columns=("ID",
"Роль", "Фамилия", "Имя",
                                                 menu items = []
"Отчество"), show="headings")
                                                  if self.can edit employees:
                                                     menu items.extend([
       for col in columns:
                                                          ("Добавить
                                         сотрудника", self.add employee),
self.employee tree.heading(col,
                                                         ("Изменить
                                          сотрудника", self.edit employee),
text=col,
                                                         ("Удалить
command=lambda
                                         сотрудника", self.delete employee),
                             c=col:
self.sort treeview(
                                                         (None, None)
                                                     ])
self.employee tree, c, False, "ID"))
                                         self.employee tree.bind("<Button-</pre>
self.employee tree.column(col,
                                                         lambda
                                                                            е:
width=100)
                                          self.create context menu(e,
                                          self.employee tree, menu items))
                                          self.employee tree.bind("<Delete>",
                                          lambda e: self.delete employee())
self.employee_tree.heading("ID",
text="ID")
                                                  self.load employees()
self.employee_tree.heading("Роль",
text="Роль")
                                             def load invoices(self):
                                                 """Загрузка данных
                                        накладных"""
self.employee tree.heading("Фамилия"
, text="Фамилия")
                                                 try:
self.employee_tree.heading("Имя",
                                         self.invoice tree.delete(*self.invoi
text="Mmg")
                                         ce tree.get children())
self.employee tree.heading("Отчество
", text="OTYECTBO")
                                         self.current search conditions['invo
                                         ice']:
                                                         conditions
self.employee_tree.column("ID",
                                         self.current search conditions['invo
width=50)
                                         ice']['conditions']
                                                         params
self.employee tree.column("Роль",
                                         self.current search conditions['invo
width=150)
                                         ice']['params']
                                                         query = """
self.employee tree.column("Фамилия",
                                                             SELECT
width=120)
                                                                 invoice id,
                                          counteragent name,
```

```
date time,
                                                     else:
                                         self.status bar.config(text=f"Наклад
type invoice text,
                                         ные загружены. Всего: {count}")
                       status text,
                       type detail,
                                                 except Exception as e:
                                                     print(f"[LOAD ERROR]
                       quantity,
                                         Failed to load invoices: {e}")
responsible last name || ' ' ||
                                         self.status bar.config(text=f"Ошибка
responsible first name || ' ' ||
                                         загрузки накладных: {str(e)}")
                                                     self.conn.rollback()
COALESCE(responsible patronymic, '')
                                             def search invoice(self):
as responsible
                                                 """Поиск накладных
                   FROM
                                         различным критериям"""
invoice details view
                   WHERE """ +
                                                 search window
AND ".join(conditions) + """
                                         Toplevel(self.root)
                   ORDER
                                 BY
                                                 search window.title("Поиск
invoice id
                                         накладных")
                                                 Label (search window,
                                         text="Критерии поиска:").grid(row=0,
self.cursor.execute(query, params)
                                         column=0, columnspan=2, pady=5)
           else:
self.cursor.execute("""
                                                Label(search window,
                                         text="ID накладной:").grid(row=1,
                                         column=0, padx=5, pady=5, sticky=W)
                       invoice id,
                                                 id var = StringVar()
                                                 id entry
counteragent name,
                       date time,
                                         Entry(search window,
                                         textvariable=id var)
                                                 id entry.grid(row=1,
type invoice text,
                       status text,
                                         column=1, padx=5, pady=5, sticky=EW)
                       type detail,
                                                 Label (search window,
                       quantity,
                                         text="Kohtparent:").grid(row=2,
responsible_last_name || ' ' ||
                                         column=0, padx=5, pady=5, sticky=W)
                                                 counteragent var
responsible first name || ' ' ||
                                         StringVar()
                                                 counteragent entry
COALESCE(responsible patronymic, '')
                                         Entry(search window,
as responsible
                                         textvariable=counteragent var)
                   FROM
invoice_details_view
                                         counteragent entry.grid(row=2,
                   ORDER
                                 BY
                                         column=1, padx=5, pady=5, sticky=EW)
invoice id
                                                 Label(search window,
                                         text="Дата
                                                                    (TTTT-MM-
                                         ДД):").grid(row=3, column=0, padx=5,
self.cursor.fetchall():
                                         pady=5, sticky=W)
                                                 date_from_var = StringVar()
self.invoice tree.insert("", END,
                                                 date_from_entry
values=row)
                                         Entry(search_window,
                                         textvariable=date_from_var)
           count
                                                 date_from_entry.grid(row=3,
len(self.invoice tree.get children()
                                         column=1, padx=5, pady=5, sticky=EW)
                                                Label(search window,
self.current search conditions['invo
                                         text="Тип накладной:").grid(row=5,
                                         column=0, padx=5, pady=5, sticky=W)
                                                 type var = StringVar()
self.status bar.config(text=f"Найден
о накладных: {count}")
```

Изм.

Лист

№ докум.

Подпись Дата

KP.350000.000

Лист 72

```
type_combobox
                                                           if
ttk.Combobox(search window,
                                          date from var.get():
textvariable=type var,
                                          conditions.append("date time >= %s")
values=["", "Отгрузка", "Выгрузка"])
        type combobox.grid(row=5,
                                          params.append(date from var.get())
column=1, padx=5, pady=5, sticky=EW)
                                                          if type var.get():
        Label (search window,
text="CTaTyc:").grid(row=6,
                                          conditions.append("type invoice text
column=0, padx=5, pady=5, sticky=W)
                                          = %s")
        status var = StringVar()
        status combobox
                                          params.append(type var.get())
ttk.Combobox(search window,
                                                          if status_var.get():
textvariable=status var,
values=["",
                         процессе",
                                          conditions.append("status text
"Завершено"])
                                          %s")
       status combobox.grid(row=6,
column=1, padx=5, pady=5, sticky=EW)
                                          params.append(status var.get())
       Label (search window,
                                                           if detail var.get():
text="Деталь:").grid(row=7,
column=0, padx=5, pady=5, sticky=W)
                                          conditions.append("type detail ILIKE
        detail var = StringVar()
                                          %s")
       detail entry
Entry(search window,
                                          params.append(f"%{detail var.get()}%
textvariable=detail var)
        detail entry.grid(row=7,
column=1, padx=5, pady=5, sticky=EW)
                                          responsible var.get():
       Label (search window,
text="Ответственный: ").grid(row=8,
                                          conditions.append("""
column=0, padx=5, pady=5, sticky=W)
                                          (responsible last name ILIKE %s OR
       responsible_var
StringVar()
                                          responsible first name ILIKE %s OR
       responsible entry
Entry(search window,
textvariable=responsible var)
                                          COALESCE(responsible patronymic, '')
                                          ILIKE %s)
responsible entry.grid(row=8,
                                                              params.extend([
column=1, padx=5, pady=5, sticky=EW)
        def perform search():
                                          f"%{responsible_var.get()}%",
            trv:
                conditions = []
                                          f"%{responsible var.get()}%",
                params = []
                                          f"%{responsible var.get()}%"
                if id var.get():
conditions.append("invoice id = %s")
                                          self.current search conditions['invo
                                          ice'] = {
params.append(int(id var.get()))
                                                               'conditions':
                                          conditions,
counteragent var.get():
                                                               'params': params
conditions.append("counteragent name
ILIKE %s")
                                                          self.load invoices()
params.append(f"%{counteragent var.g search window.destroy()
et()}%")
                                                      except ValueError as ve:
```

11 11 11

```
messagebox.showerror("Ошибка",
f"Некорректные данные: {str(ve)}")
                                         self.cursor.execute(query, params)
            except Exception as e:
                                                      else:
                                         self.cursor.execute("""
self.status_bar.config(text=f"Ошибка
поиска: {str(e)}")
                                                              SELECT
                self.conn.rollback()
                                                                  detail id,
        def reset search():
                                          warehouse number,
                                                                  room number,
self.current search conditions['invo
                                                                  rack number,
ice'] = None
            self.load invoices()
                                          shelf number,
            search window.destroy()
                                                                  type detail,
                                                                  weight
       Button (search window,
text="Найти",
                                         warehouse details view
command=perform search).grid(
                                                             ORDER
            row=9, column=0, padx=5,
                                         detail id,
                                                            warehouse number,
pady=10, sticky=EW)
                                         room number,
                                                                 rack number,
       Button(search window,
                                         shelf number
text="Сбросить",
                                                          """)
command=reset search).grid(
            row=9, column=1, padx=5,
                                                      for
                                                                            in
                                                                row
                                         self.cursor.fetchall():
pady=10, sticky=EW)
                                          self.warehouse_tree.insert("", END,
    def load warehouse(self):
        """Загрузка
                      данных
                                         values=row)
складе"""
                                                      count
       try:
                                          len(self.warehouse tree.get children
self.warehouse tree.delete(*self.war
                                          ())
ehouse_tree.get_children())
                                          self.current_search_conditions['ware
                                          house']:
self.current_search_conditions['ware
house']:
                                          self.status bar.config(text=f"Найден
                                          о деталей: {count}")
               conditions
self.current search conditions['ware
                                                      else:
house']['conditions']
               params
                                          self.status bar.config(text=f"Данные
self.current search conditions['ware
                                          склада загружены. Всего: {count}")
house']['params']
                                                 except Exception as e:
               query = """
                                                      print(f"[LOAD ERROR]
                   SELECT
                                          Failed to load warehouse data: {e}")
                        detail id,
                                          self.status bar.config(text=f"Ошибка
warehouse number,
                                          загрузки данных склада: {str(e)}")
                        room number,
                                                      self.conn.rollback()
                        rack number,
                                             def search_warehouse_item(self):
                                                  """Поиск деталей на складе
shelf number,
                                          по различным критериям"""
                        type detail,
                        weight
                                                 search window
                    FROM
                                          Toplevel(self.root)
warehouse_details view
                                                 search window.title("Поиск
                    WHERE """ + "
                                          деталей на складе")
AND ".join(conditions) + """
                   ORDER
                                                  Label (search window,
                                BY
detail id,
                  warehouse number,
                                          text="Критерии поиска:").grid(row=0,
room number,
                                          column=0, columnspan=2, pady=5)
                       rack number,
shelf number
```

Изм. Лист № докум. Подпись Дата

```
Label(search_window,
       Label (search window,
text="Тип детали:").grid(row=1,
                                         text="Bec
                                                            до:").grid(row=7,
column=0, padx=5, pady=5, sticky=W)
                                         column=0, padx=5, pady=5, sticky=W)
       type_var = StringVar()
                                                 weight_to_var = StringVar()
        type entry
                                                  weight to entry
Entry (search window,
                                          Entry(search window,
textvariable=type var)
                                          textvariable=weight to var)
        type_entry.grid(row=1,
                                                  weight_to_entry.grid(row=7,
column=1, padx=5, pady=5, sticky=EW)
                                          column=1, padx=5, pady=5, sticky=EW)
        Label (search window,
                                                  def perform search():
text="Homep склада:").grid(row=2,
column=0, padx=5, pady=5, sticky=W)
                                                          conditions = []
       warehouse var = StringVar()
                                                          params = []
       warehouse entry
Entry(search window,
                                                          if type var.get():
textvariable=warehouse var)
       warehouse entry.grid(row=2,
                                          conditions.append("type detail ILIKE
column=1, padx=5, pady=5, sticky=EW)
       Label (search window,
                                          params.append(f"%{type var.get()}%")
text="Homep комнаты:").grid(row=3,
column=0, padx=5, pady=5, sticky=W)
       room var = StringVar()
                                         warehouse var.get():
       room entry
Entry(search window,
                                          conditions.append("warehouse number
textvariable=room var)
                                          = %s")
       room entry.grid(row=3,
column=1, padx=5, pady=5, sticky=EW)
                                         params.append(warehouse var.get())
       Label (search window,
                                                          if room var.get():
text="Homep стеллажа:").grid(row=4,
                                         conditions.append("room number
column=0, padx=5, pady=5, sticky=W)
       rack var = StringVar()
                                         %s")
       rack entry
Entry(search_window,
                                          params.append(room_var.get())
textvariable=rack_var)
       rack_entry.grid(row=4,
                                                          if rack var.get():
column=1, padx=5, pady=5, sticky=EW)
                                          conditions.append("rack number
                                          %s")
       Label(search window,
text="Homep полки:").grid(row=5,
column=0, padx=5, pady=5, sticky=W)
                                         params.append(rack var.get())
       shelf_var = StringVar()
       shelf_entry
                                                          if shelf var.get():
Entry(search window,
textvariable=shelf_var)
                                          conditions.append("shelf number
       shelf entry.grid(row=5,
column=1, padx=5, pady=5, sticky=EW)
                                          params.append(shelf var.get())
       Label(search window,
text="Bec
                 от:").grid(row=6,
column=0, padx=5, pady=5, sticky=W)
                                          weight from var.get():
       weight_from_var
                                                                  weight from
StringVar()
       weight_from_entry
                                         = float(weight from var.get())
Entry(search window,
textvariable=weight from var)
                                          conditions.append("weight >= %s")
weight from entry.grid(row=6,
                                          params.append(weight from)
column=1, padx=5, pady=5, sticky=EW)
                                                              except
                                          ValueError:
```

Изм.	Лист	№ докум.	Подпись	Дато

```
messagebox.showwarning("Предупрежден self.counteragent_tree.delete(*self.
ие", "Некорректное значение веса
                                        counteragent tree.get children())
'OT'")
               if
                                         self.current search conditions['coun
weight to var.get():
                                         teragent']:
                                                         conditions
                   trv:
                       weight to =
                                    self.current search conditions['coun
float(weight_to_var.get())
                                        teragent']['conditions']
                                                        params
                                         self.current_search_conditions['coun
conditions.append("weight <= %s")</pre>
                                         teragent']['params']
                                                        query = "SELECT *
params.append(weight to)
                                         FROM counteragent WHERE " + " AND
                   except
                                         ".join(conditions) + " ORDER BY
ValueError:
                                         counteragent id"
messagebox.showwarning("Предупрежден
ие", "Некорректное значение веса
                                        self.cursor.execute(query, params)
'до'")
                                         self.cursor.execute("SELECT * FROM
self.current search conditions['ware
                                                            ORDER
                                         counteragent
                                                                           BY
house'] = {
                                         counteragent id")
                   'conditions':
conditions,
                                                     for
                                                                          in
                                                               row
                                        self.cursor.fetchall():
                   'params': params
                                         self.counteragent tree.insert("",
                                         END, values=row)
self.load warehouse()
                                                     count
search window.destroy()
                                         len(self.counteragent tree.get child
                                         ren())
           except Exception as e:
                                         self.current_search_conditions['coun
                                         teragent']:
self.status bar.config(text=f"Ошибка
поиска: {str(e)}")
               self.conn.rollback()
                                         self.status bar.config(text=f"Найден
                                         o контрагентов: {count}")
        def reset search():
                                                     else:
self.current search conditions['ware
                                         self.status bar.config(text=f"Контра
house'] = None
                                         генты загружены. Всего: {count}")
           self.load warehouse()
                                                except Exception as e:
           search window.destroy()
                                         self.status bar.config(text=f"Ошибка
       Button(search window,
                                         : {str(e)}")
text="Найти",
command=perform search).grid(
                                             def search counteragent(self):
           row=8, column=0, padx=5,
                                                """Поиск контрагентов по
                                         различным критериям"""
pady=10, sticky=EW)
       Button(search window,
                                                search window
text="Cбросить",
                                         Toplevel(self.root)
command=reset search).grid(
                                                search window.title("Поиск
           row=8, column=1, padx=5,
                                         контрагентов")
pady=10, sticky=EW)
                                                 Label(search_window,
                                         text="Критерии поиска:").grid(row=0,
    def load_counteragents(self):
                                         column=0, columnspan=2, pady=5)
       """Загрузка данных
                                 0
контрагентах"""
       try:
```

Изм.

Лист

№ докум.

Подпись Дата

KP.350000.000

```
Label (search window,
                                       conditions.append("counteragent name
text="ID контрагента:").grid(row=1,
column=0, padx=5, pady=5, sticky=W)
                                         ILIKE %s")
        id var = StringVar()
        {\tt id\_entry}
                                         params.append(f"%{name var.get()}%")
Entry(search window,
textvariable=id var)
        id_entry.grid(row=1,
                                         contact var.get():
column=1, padx=5, pady=5, sticky=EW)
                                          conditions.append("contact person
        Label (search window,
                                          ILIKE %s")
text="Haзвание:").grid(row=2,
column=0, padx=5, pady=5, sticky=W)
                                         params.append(f"%{contact var.get()}
        name var = StringVar()
                                          용")
       name entry
Entry(search window,
                                                          if phone var.get():
textvariable=name var)
       name entry.grid(row=2,
                                          conditions.append("phone number::tex
column=1, padx=5, pady=5, sticky=EW)
                                         t LIKE %s")
       Label (search window,
                                          params.append(f"%{phone var.get()}%"
text="Контактное лицо:").grid(row=3,
column=0, padx=5, pady=5, sticky=W)
       contact var = StringVar()
       contact entry
                                          address var.get():
Entry(search window,
textvariable=contact var)
                                          conditions.append("address
                                                                         ILIKE
       contact entry.grid(row=3,
column=1, padx=5, pady=5, sticky=EW)
                                          params.append(f"%{address var.get()}
       Label (search window,
                                          응배)
text="Телефон:").grid(row=4,
column=0, padx=5, pady=5, sticky=W)
       phone var = StringVar()
                                         self.current search conditions['coun
       phone entry
                                         teragent'] = {
Entry(search_window,
                                                              'conditions':
textvariable=phone_var)
                                         conditions,
                                                              'params': params
       phone entry.grid(row=4,
column=1, padx=5, pady=5, sticky=EW)
                                                          }
       Label (search window,
text="Aдpec:").grid(row=5, column=0,
                                         self.load counteragents()
padx=5, pady=5, sticky=W)
       address_var = StringVar()
                                         search window.destroy()
       address_entry
Entry(search window,
                                                      except ValueError as ve:
textvariable=address var)
       address entry.grid(row=5,
                                         messagebox.showerror("Ошибка",
column=1, padx=5, pady=5, sticky=EW)
                                         f"Некорректные данные: {str(ve)}")
                                                      except Exception as e:
        def perform search():
            trv:
                                         self.status bar.config(text=f"Ошибка
                conditions = []
                                         поиска: {str(e)}")
                params = []
                                                         self.conn.rollback()
                if id_var.get():
                                                  def reset search():
conditions.append("counteragent id =
                                         self.current search conditions['coun
%s")
                                          teragent'] = None
params.append(int(id var.get()))
                                         self.load counteragents()
                                                      search window.destroy()
                if name var.get():
```

Изм.	Лист	№ докум.	Подпись	Дата

```
Button (search window,
                                              def search employee(self):
text="Найти",
                                                  """Поиск сотрудников
                                                                            ПО
command=perform search).grid(
                                          различным критериям"""
            row=6, column=0, padx=5,
                                                 search window
pady=10, sticky=EW)
                                          Toplevel(self.root)
        Button (search window,
                                                  search window.title("Поиск
text="Сбросить",
                                          сотрудников")
command=reset search).grid(
            row=6, column=1, padx=5,
                                                  Label (search window,
                                          text="Критерии поиска:").grid(row=0,
pady=10, sticky=EW)
                                          column=0, columnspan=2, pady=5)
    def load employees(self):
        """Загрузка
                                                  Label (search window,
                       данных
сотрудниках"""
                                          text="ID сотрудника:").grid(row=1,
                                          column=0, padx=5, pady=5, sticky=W)
       try:
                                                  id var = StringVar()
self.employee tree.delete(*self.empl
                                                  id entry
oyee tree.get children())
                                          Entry (search window,
                                          textvariable=id var)
                                                  id entry.grid(row=1,
self.current search conditions['empl
                                          column=1, padx=5, pady=5, sticky=EW)
               conditions
                                                  Label (search window,
self.current search conditions['empl
                                          text="Роль:").grid(row=2, column=0,
oyee']['conditions']
                                          padx=5, pady=5, sticky=W)
                                                  role var = StringVar()
               params
self.current search conditions['empl
                                                  role combobox
oyee']['params']
                                          ttk.Combobox(search window,
                query = "SELECT *
                                          textvariable=role var,
FROM employee WHERE " + " AND
".join(conditions) + " ORDER BY
                                          values=["", "Кладовщик", "Менеджер
                                          склада", "Владелец"])
employee id"
                                                 role combobox.grid(row=2,
                                          column=1, padx=5, pady=5, sticky=EW)
self.cursor.execute(query, params)
            else:
                                                  Label (search window,
self.cursor.execute("SELECT * FROM
                                          text="Фамилия:").grid(row=3,
                                          column=0, padx=5, pady=5, sticky=W)
employee ORDER BY employee id")
                                                  last name var = StringVar()
                                                  last name entry
            for
                      row
                                 in
self.cursor.fetchall():
                                          Entry(search window,
                                          textvariable=last_name_var)
self.employee_tree.insert("",
                                END,
                                                  last_name_entry.grid(row=3,
values=row)
                                          column=1, padx=5, pady=5, sticky=EW)
                                                  Label (search window,
len(self.employee tree.get children(
                                          text="Имя:").grid(row=4,
                                                                     column=0,
                                          padx=5, pady=5, sticky=W)
                                                  first_name_var = StringVar()
self.current search conditions['empl
                                                  first name entry
oyee']:
                                          Entry(search window,
                                          textvariable=first_name_var)
self.status bar.config(text=f"Найден
                                                  first_name_entry.grid(row=4,
o сотрудников: {count}")
                                          column=1, padx=5, pady=5, sticky=EW)
           else:
                                                  Label(search_window,
self.status bar.config(text=f"Сотруд
                                          text="Отчество:").grid(row=5,
ники загружены. Всего: {count}")
                                          column=0, padx=5, pady=5, sticky=W)
       except Exception as e:
                                                  patronymic_var = StringVar()
                                                  patronymic entry
self.status bar.config(text=f"Ошибка
                                          Entry(search window,
: {str(e)}")
                                          textvariable=patronymic var)
```

Изм.	Лист	№ докум.	Подпись	Дата

```
patronymic entry.grid(row=5,
column=1, padx=5, pady=5, sticky=EW)
                                          messagebox.showerror("Ошибка",
                                          f"Некорректные данные: {str(ve)}")
        def perform search():
                                                      except Exception as e:
            try:
                conditions = []
                                          self.status bar.config(text=f"Ошибка
                                          поиска: {str(e)}")
                params = []
                                                          self.conn.rollback()
                if id var.get():
                                                  def reset search():
conditions.append("employee id
%s")
                                          self.current search conditions['empl
                                          oyee'] = None
params.append(int(id var.get()))
                                                      self.load employees()
                                                      search window.destroy()
                if role var.get():
                                                  Button (search window,
conditions.append("employee role =
                                          text="Найти",
                                          command=perform search).grid(
                                                      row=6, column=0, padx=5,
params.append(role var.get())
                                          pady=10, sticky=EW)
                                                  Button(search window,
                                          text="Cбросить",
                                          command=reset search).grid(
last name var.get():
                                                      row=6, column=1, padx=5,
conditions.append("last name
                               ILIKE
                                          pady=10, sticky=EW)
                                              def add invoice(self):
                                                  """Добавление
params.append(f"%{last name var.get(
                                                                         новой
                                          накладной с проверкой прав"""
) } 응")
                                                  i f
                                                                           not
                                          self.can edit invoices:
first name var.get():
                                          messagebox.showerror("Ошибка",
conditions.append("first name ILIKE
                                          вас нет
                                                       прав на добавление
                                          накладных")
                                                      return
params.append(f"%{first_name_var.get
()}%")
                                                      add window
                                          Toplevel(self.root)
patronymic var.get():
                                          add window.title("Добавить
conditions.append("patronymic ILIKE
                                          накладную")
%s")
params.append(f"%{patronymic var.get
                                          self.can_access_counteragents:
                                          self.cursor.execute("SELECT
                                          counteragent id, counteragent name
                                          FROM counteragent")
self.current search conditions['empl
oyee'] = {
                                                          counteragents
                    'conditions':
                                          self.cursor.fetchall()
conditions,
                                                          counteragent names =
                    'params': params
                                         [name for id, name in counteragents]
                }
                                                          counteragent ids
                                          {name:
                                                  id
                                                       for id,
                                                                   name
                                                                            in
                                          counteragents}
self.load employees()
                                                      else:
                                                          counteragents = []
search window.destroy()
                                                          counteragent names =
                                          []
            except ValueError as ve:
```

```
textvariable=type_var,
                counteragent ids =
                                          values=["Отгрузка", "Выгрузка"])
{ }
                                                      type_combobox.current(0)
messagebox.showwarning("Предупрежден
     "Нет
             доступа к списку
                                          type combobox.grid(row=2, column=1,
                                          padx=5, pady=5, sticky=EW)
контрагентов")
            self.cursor.execute("""
                                                      Label (add window,
                                         text="CTaTyc:").grid(row=3,
                SELECT DISTINCT
                    responsible id,
                                          column=0, padx=5, pady=5, sticky=W)
                                                      status var = StringVar()
responsible_last_name || ' ' ||
                                                      status combobox
                                          ttk.Combobox(add window,
responsible first name || ' ' ||
                                          textvariable=status var, values=["B
                                          процессе", "Завершено"])
COALESCE (responsible patronymic, '')
as responsible name
                                          status combobox.current(0)
invoice_details_view
                                          status combobox.grid(row=3,
                ORDER
                                 BY
                                          column=1, padx=5, pady=5, sticky=EW)
responsible name
                                                      detail frame
                                          Frame(add window)
            employees
                                                      detail frame.grid(row=4,
self.cursor.fetchall()
           employee names = [name
                                          column=0,
                                                      columnspan=2, padx=5,
for id, name in employees]
                                          pady=5, sticky=EW)
           employee ids = {name: id
for id, name in employees}
                                                     Label (detail frame,
                                          text="Деталь:").pack(side=LEFT,
           Label(add window,
                                          padx = (0, 5)
text="Контрагент:").grid(row=0,
                                                      detail var = StringVar()
column=0, padx=5, pady=5, sticky=W)
                                                      detail entry
            counteragent var
                                          Entry(detail frame,
                                          textvariable=detail var)
StringVar()
            counteragent_combobox =
ttk.Combobox(add window,
                                          detail entry.pack(side=LEFT,
                                          expand=True, fill=X)
textvariable=counteragent var,
values=counteragent names)
                                                      search btn
counteragent combobox.grid(row=0,
                                                                   text="\\",
                                          Button(detail frame,
column=1, padx=5, pady=5, sticky=EW)
                                          command=lambda:
                                          self.search detail(detail var))
           Label (add window,
text="Дата
           и время (ГГГГ-ММ-ДД
                                          search_btn.pack(side=LEFT, padx=(5,
                         column=0,
ЧЧ:MM):").grid(row=1,
padx=5, pady=5, sticky=W)
            date entry
                                                      Label (add window,
Entry(add window)
                                          text="Количество:").grid(row=5,
           current datetime
                                          column=0, padx=5, pady=5, sticky=W)
datetime.now().strftime("%Y-%m-%d
                                                      quantity entry
%H:%M")
                                         Entry(add window)
           date entry.insert(0,
current datetime)
                                          quantity entry.grid(row=5, column=1,
            date_entry.grid(row=1,
                                          padx=5, pady=5, sticky=EW)
column=1, padx=5, pady=5, sticky=EW)
                                                      Label (add window,
           Label (add window,
                                          text="Ответственный:").grid(row=6,
text="Тип
           накладной:").grid(row=2,
                                          column=0, padx=5, pady=5, sticky=W)
column=0, padx=5, pady=5, sticky=W)
                                                      employee_var
           type var = StringVar()
                                          StringVar()
            type combobox
                                                      employee_combobox
ttk.Combobox(add window,
                                          ttk.Combobox(add window,
```

KP.350000.000

```
textvariable=employee_var,
                                                               if
                                                                    None
                                                                             in
values=employee names)
                                           (counteragent id, employee id):
                                                                   raise
employee_combobox.grid(row=6,
                                           ValueError("Не все обязательные поля
column=1, padx=5, pady=5, sticky=EW)
                                          заполнены")
            def save invoice():
                                                               detail name
                                          detail var.get().strip()
                try:
                                                                            not.
input datetime str
                                          detail name:
date entry.get()
                                                                   raise
                                           ValueError ("Название детали не может
                    try:
                                           быть пустым")
input datetime
datetime.strptime(input_datetime_str
                                          self.cursor.execute("""
, "%Y-%m-%d %H:%M")
                    except
                                                                   SELECT
ValueError:
                                          detail id FROM details
                                                                   WHERE
ValueError("Некорректный
                                           type detail = %s
                              формат
даты. Используйте ГГГГ-ММ-ДД ЧЧ:ММ")
                                                                   LIMIT 1
                                                               """,
                                           (detail name,))
                    current datetime
= datetime.now()
                                                               detail data
input datetime > current datetime:
                                           self.cursor.fetchone()
                        raise
                                                                            not
ValueError("Дата накладной не может
                                           detail data:
быть в будущем. Укажите текущую или
                                                                   raise
                                          ValueError(f"Деталь '{detail name}'
прошедшую дату.")
                                          не найдена на складе")
                    counteragent id
                                                               detail id
                                          detail data[0]
counteragent ids.get(counteragent va
r.get())
                    employee id
                                                               type_invoice
                                          type_var.get() == "Выгрузка"
employee_ids.get(employee_var.get())
                                                               status
                                           status_var.get() == "Завершено"
                    if employee id:
self.cursor.execute("""
                            SELECT
                                                                   quantity
COUNT(*) FROM invoice employee
                                          int(quantity entry.get())
                                                                   if quantity
responsible = %s AND invoiceid IN (
                                          <= 0:
                                                                       raise
SELECT invoice id FROM invoice WHERE
                                          ValueError("Количество должно быть
status = FALSE
                                          положительным числом")
                                                               except
                                          ValueError:
(employee id,))
                                                                   raise
                                          ValueError("Количество должно быть
active invoices count
                                          целым числом")
self.cursor.fetchone()[0]
                                                               if
                                                                            not
                                          type invoice:
active invoices_count >= 5:
                            raise
                                          self.cursor.execute("""
ValueError(f"У
                 сотрудника уже
                                                                       SELECT
{active invoices count}
                                          COUNT (*)
                            активных
накладных. Максимум - 5.")
                                                                       FROM
                                           details
```

Изм.	Лист	№ докум.	Подпись	Дата

```
WHERE
                                                                INSERT INTO
type detail = %s
                                        invoice detail (invoiceid, detailid,
                                        quantity)
(detail name,))
                                                                VALUES (%s,
                                        %s, %s)
total available
self.cursor.fetchone()[0]
                                         (invoice id, detail id, quantity))
self.cursor.execute("""
                                        self.cursor.execute("""
                           SELECT
                                                               INSERT INTO
COALESCE(SUM(id.quantity), 0)
                                        invoice employee
                                                                (invoiceid,
                           FROM
                                        responsible,
                                                             granted access,
invoice detail id
                                        when granted)
                           JOIN
                                                               VALUES (%s,
invoice i ON id.invoiceid =
                                        %s, %s, NOW())
i.invoice id
                           JOIN
                                         (invoice id,
                                                                employee id,
details d
              ON
                 id.detailid
                                        employee id))
d.detail id
d.type detail =
                       %s AND
                                        self.conn.commit()
i.type invoice = FALSE AND i.status
= FALSE
                                        self.load invoices()
                       """,
(detail name,))
                                        add window.destroy()
                       reserved
self.cursor.fetchone()[0]
                                        messagebox.showinfo("Успех",
                                         "Накладная успешно добавлена")
                       i f
                                                        except ValueError as
total available < (reserved
                                        ve:
quantity):
                                        messagebox.showerror("Ошибка",
                           raise
ValueError(
                                        f"Неверные данные: {str(ve)}")
                                                       except Exception as
f"Недостаточно деталей на складе для
отгрузки.
                          Доступно:
{total_available}, "
                                        self.conn.rollback()
f"уже зарезервировано для отгрузки:
                                        messagebox.showerror("Ошибка", f"He
{reserved}, требуется: {quantity}"
                                        удалось
                                                   добавить накладную:
                                         {str(e)}")
                                                    Button (add window,
self.cursor.execute("""
                                        text="Сохранить",
                       INSERT INTO
                                        command=save invoice).grid(row=7,
invoice (counteragentid, date time,
                                        column=0, columnspan=2, pady=10)
type invoice, status)
                       VALUES (%s,
                                                except Exception as e:
%s, %s, %s)
                                        messagebox.showerror("Ошибка", f"He
                       RETURNING
invoice id
                                        удалось открыть форму: {str(e)}")
(counteragent id,
                                            def edit invoice(self):
                                                """Редактирование накладной
input_datetime_str, type_invoice,
                                        с проверкой прав"""
status))
                                                if
                                                                         not
                   invoice id
                                        self.can edit invoices:
self.cursor.fetchone()[0]
                                        messagebox.showerror("Ошибка",
                                        вас нет прав на редактирование
                                        накладных")
self.cursor.execute("""
                                                    return
```

Изм.	Лист	№ докум.	Подпись	Дата

```
counteragent ids
                                        {name: id for id, name
       selected
                                                                           in
self.invoice tree.selection()
                                         counteragents }
       if not selected:
                                                     self.cursor.execute("""
messagebox.showwarning("Предупрежден
                                                         SELECT DISTINCT
ие", "Выберите накладную для
                                                             responsible id,
редактирования")
                                         responsible last name || ' ' ||
           return
                                         responsible first name || ' ' ||
        item
self.invoice tree.item(selected[0])
       invoice id
                                         COALESCE(responsible patronymic, '')
item['values'][0]
                                         as responsible name
                                                         FROM
                                         invoice_details_view
            self.cursor.execute("""
                                                                           BY
                                         responsible_name
               SELECT
                   invoice id,
                                                     employees
                                         self.cursor.fetchall()
counteragent name,
                   date time,
                                                     employee names = [name
                                         for id, name in employees]
                                                    employee ids = {name: id
type invoice text,
                   status text,
                                         for id, name in employees}
                   type detail,
                   quantity,
                                                     Label (edit window,
                                         text="Kohtparent:").grid(row=0,
                                         column=0, padx=5, pady=5, sticky=W)
responsible last name || ' ' ||
                                                     counteragent var
responsible first name || ' ' ||
                                        StringVar()
                                                     counteragent combobox =
COALESCE (responsible patronymic, '')
                                        ttk.Combobox(edit window,
as responsible
                                         textvariable=counteragent var,
                FROM
invoice_details_view
                                         values=counteragent names)
               WHERE invoice id =
                                         counteragent combobox.grid(row=0,
            """, (invoice_id,))
                                         column=1, padx=5, pady=5, sticky=EW)
            invoice data
                                                     for id,
                                                                  name
self.cursor.fetchone()
                                        counteragents:
                                                         if
                                                                name
           if not invoice data:
                                         invoice data[1]:
messagebox.showerror("Ошибка",
                                         counteragent var.set(invoice data[1]
"Накладная не найдена")
               return
                                                             break
           edit window
                                                     Label(edit window,
                                         text="Дата и время:").grid(row=1,
Toplevel(self.root)
                                         column=0, padx=5, pady=5, sticky=W)
edit window.title("Редактировать
                                                     date entry
накладную")
                                         Entry(edit window)
                                                     date entry.insert(0,
                                         invoice data[2].strftime("%Y-%m-%d
self.cursor.execute("SELECT
                                         %H:%M"))
counteragent id, counteragent name
                                                     date entry.grid(row=1,
FROM counteragent")
                                         column=1, padx=5, pady=5, sticky=EW)
           counteragents
self.cursor.fetchall()
                                                     Label (edit window,
           counteragent_names =
                                         text="Тип накладной:").grid(row=2,
[name for id, name in counteragents]
                                         column=0, padx=5, pady=5, sticky=W)
```

Изм.

Лист

№ докум.

Подпись

Дата

KP.350000.000

```
type var = StringVar()
            type combobox
                                         quantity entry.grid(row=5, column=1,
ttk.Combobox(edit window,
                                          padx=5, pady=5, sticky=EW)
textvariable=type_var,
                                                      Label (edit window,
values=["Отгрузка", "Выгрузка"])
                                          text="Ответственный:").grid(row=6,
            type combobox.current(1
                                          column=0, padx=5, pady=5, sticky=W)
if invoice data[3] == "Выгрузка"
                                                      employee var
else 0)
                                          StringVar()
                                                      employee combobox
type combobox.grid(row=2, column=1,
                                          ttk.Combobox(edit window,
padx=5, pady=5, sticky=EW)
                                          textvariable=employee var,
            Label (edit window,
                                         values=employee names)
text="CTaTyc:").grid(row=3,
column=0, padx=5, pady=5, sticky=W)
                                          employee combobox.grid(row=6,
            status var = StringVar()
                                          column=1, padx=5, pady=5, sticky=EW)
            status combobox
ttk.Combobox(edit window,
                                                      for id,
textvariable=status var,
                                          employees:
                                                                 name
values=["В процессе", "Завершено"])
                                          invoice data[7]:
status combobox.current(1
                                          employee var.set(name)
invoice data[4] == "Завершено" else
                                                              break
                                                      def save changes():
status combobox.grid(row=3,
                                                          try:
column=1, padx=5, pady=5, sticky=EW)
                                                              detail name
                                         detail var.get().strip()
           detail frame
                                                                           not
Frame(edit window)
                                          detail name:
           detail frame.grid(row=4,
                                                                  raise
            columnspan=2, padx=5,
                                          ValueError("Тип детали не может быть
column=0,
pady=5, sticky=EW)
                                          пустым")
           Label (detail frame,
                                          self.cursor.execute("""
text="Деталь:").pack(side=LEFT,
                                                                  SELECT
padx = (0, 5)
            detail var = StringVar()
                                         detail id FROM details
            detail entry =
                                                                  WHERE
Entry(detail frame,
                                          type detail = %s
textvariable=detail var)
                                                                  LIMIT 1
detail entry.pack(side=LEFT,
                                          (detail name,))
expand=True, fill=X)
                                                              detail data
            search btn
                                          self.cursor.fetchone()
                         text="\",
                                                              if
                                                                           not
Button (detail frame,
                                          detail data:
command=lambda:
                                                                  raise
self.search detail(detail var))
                                          ValueError(f"Деталь '{detail name}'
                                         не найдена на складе")
search btn.pack(side=LEFT, padx=(5,
                                                              detail id
                                          detail data[0]
           Label (edit window,
text="Количество:").grid(row=5,
                                                              trv:
column=0, padx=5, pady=5, sticky=W)
                                                                  quantity
            quantity_entry
                                          int(quantity_entry.get())
Entry(edit window)
                                                                  if quantity
            quantity_entry.insert(0,
                                          <= 0:
str(invoice data[6]))
```

 Изм.
 Лист
 № докум.
 Подпись
 Дата

KP.350000.000

Лист **Q** 1

```
raise
ValueError("Количество должно быть
                                                             if
                                                                          not
положительным числом!")
                                         type invoice:
                   except
ValueError:
                                         self.cursor.execute("""
                       raise
                                                                     SELECT
ValueError ("Количество должно быть
                                         quantity FROM invoice detail
целым числом!")
                                                                     WHERE
                                         invoiceid = %s
                                                                 """,
                                         (invoice id,))
counteragent name
counteragent var.get()
                   counteragent id
                                         old quantity row
                                         self.cursor.fetchone()
counteragent ids.get(counteragent na
                                                                 old quantity
                                                 old quantity row[0]
                                         old quantity row else 0
                   if
counteragent_id is None:
                                         self.cursor.execute("""
                       raise
ValueError("Выберите
                                                                     SELECT
                       корректного
контрагента")
                                         COUNT(*)
                                                                     FROM
                   employee name =
                                         details d
employee var.get()
                                                                     WHERE
                                         d.type_detail = %s
                   employee id
employee ids.get(employee name)
                                                                     AND NOT
                                         EXISTS (
                   if employee id
is None:
                                         SELECT 1 FROM invoice detail id
                       raise
                                                                         JOIN
ValueError("Выберите
                                         invoice i
                                                        ON
                                                           id.invoiceid
                       корректного
сотрудника")
                                         i.invoice id
                   if employee id:
                                         WHERE id.detailid = d.detail id
self.cursor.execute("""
                                         i.type_invoice = FALSE --- отгрузка
                           SELECT
                                                                         AND
                                         i.status = TRUE --- завершённые
COUNT(*) FROM invoice employee
                           WHERE
responsible = %s
                                         (detail name,))
                           AND
invoiceid IN (SELECT invoice id FROM
invoice WHERE status = FALSE)
                                         total available
                           AND
                                         self.cursor.fetchone()[0]
invoiceid != %s
                                         self.cursor.execute("""
(employee id, invoice id))
active invoices count
                                         COALESCE(SUM(id.quantity), 0)
self.cursor.fetchone()[0]
                                         invoice detail id
                                                                     JOIN
active_invoices_count >= 5:
                                         invoice i
                                                       ON id.invoiceid
                           raise
                                         i.invoice id
ValueError(f"У сотрудника уже
                                                                     JOIN
                                         details d ON
                                                             id.detailid
{active invoices count} активных
накладных. Максимум - 5.")
                                         d.detail id
                                                                     WHERE
                                         d.type detail = %s
                   type invoice
type var.get() == "Выгрузка"
                                                                     AND
                   status
                                         i.type invoice = FALSE --- отгрузка
status var.get() == "Завершено"
```

№ докум.

Подпись

Дата

Изм.

Лист

KP.350000.000

```
i.status = FALSE --- в процессе
                                        self.conn.commit()
i.invoice id != %s
                                         self.load invoices()
(detail name, invoice id))
                                         edit window.destroy()
                       reserved
self.cursor.fetchone()[0]
                                         messagebox.showinfo("Успех",
                                         "Накладная успешно обновлена")
                       if
                                                        except ValueError as
total available < (reserved
                                         ve:
quantity - old quantity):
                                        messagebox.showerror("Ошибка",
                           raise
ValueError(
                                         f"Hеверные данные: {str(ve)}")
                                                        except Exception as
f"Недостаточно деталей на складе для
отгрузки. "
                                         self.conn.rollback()
f"Доступно: {total_available}, "
                                         messagebox.showerror("Ошибка", f"He
f"уже зарезервировано в других
                                         удалось
                                                    обновить накладную:
накладных: {reserved}, "
                                         {str(e)}")
f"новое
         количество:
                        {quantity}
                                                    Button(edit window,
(было: {old_quantity})"
                                         text="Сохранить",
                                         command=save changes).grid(row=7,
                                         column=0, columnspan=2, pady=10)
self.cursor.execute("""
                                                 except Exception as e:
                       UPDATE
invoice
                                         messagebox.showerror("Ошибка", f"He
                       SET
                                         удалось открыть форму: {str(e)}")
counteragentid = %s, date time = %s,
type invoice = %s, status = %s
                                            def delete invoice(self):
                                                 """Удаление накладной
                       WHERE
invoice_id = %s
                                         предварительным удалением связанных
                                         записей"""
(counteragent id, date entry.get(),
                                                selected
type invoice, status, invoice id))
                                         self.invoice tree.selection()
                                                if not selected:
self.cursor.execute("""
                                         messagebox.showwarning("Предупрежден
                                         ие", "Выберите накладную для
                       UPDATE
invoice detail
                                         удаления")
                       SET detailid
                                                    return
= %s, quantity = %s
                       WHERE
invoiceid = %s
                                         self.invoice tree.item(selected[0])
                   """, (detail id,
                                                invoice id
quantity, invoice id))
                                         item['values'][0]
self.cursor.execute("""
                                         messagebox.askyesno("Подтверждение",
                       UPDATE
                                         f"Вы уверены, что хотите удалить
invoice employee
                                         накладную №{invoice id}?"):
                       SET
                                                    try:
responsible = %s
                       WHERE
                                        self.cursor.execute("DELETE
                                                                       FROM
invoiceid = %s
                                         invoice detail WHERE invoiceid =
                                         %s", (invoice id,))
(employee id, invoice id))
                                         self.cursor.execute("DELETE
                                                                       FROM
```

```
invoice employee WHERE invoiceid =
                                                    warehouse options
%s", (invoice id,))
                                        [str(number) for id, number
                                                                          in
                                        warehouses]
                                                    warehouse_ids
self.cursor.execute("DELETE FROM
                                      {str(number): id for id, number in
                                        warehouses}
invoice WHERE invoice id = %s",
(invoice id,))
                                                    row = 0
               self.conn.commit()
               self.load invoices()
                                                    Label (add window,
                                         text="Homep склада:").grid(row=row,
                                         column=0, padx=5, pady=5, sticky=W)
               if
len(self.invoice_tree.get_children()
                                                    warehouse var
                0
                                         StringVar()
         ===
self.current search conditions['invo
                                                     warehouse combobox
                                         ttk.Combobox(add window,
                                         textvariable=warehouse var,
messagebox.showinfo("Информация",
"Таблица
         пуста. Сброс условий
                                         values=warehouse options,
поиска.")
                                         state="readonly")
self.current search conditions['invo
                                         warehouse combobox.grid(row=row,
ice'] = None
                                         column=1, padx=5, pady=5, sticky=EW)
                                                    row += 1
self.load invoices()
                                                    Label (add window,
                                         text="Homep комнаты:").grid(row=row,
                                         column=0, padx=5, pady=5, sticky=W)
messagebox.showinfo("Успех",
                                                    room var = StringVar()
"Накладная успешно удалена")
           except Exception as e:
                                                     room combobox
               self.conn.rollback()
                                        ttk.Combobox(add window,
                                         textvariable=room var,
messagebox.showerror("Ошибка", f"He
удалось удалить накладную:
                                         values=[], state="readonly")
{str(e)}")
                                         room combobox.grid(row=row,
    def add_warehouse_item(self):
                                         column=1, padx=5, pady=5, sticky=EW)
       """Добавление детали на
                                                    row += 1
склад с динамическими выпадающими
списками"""
                                                     Label (add window,
                                         text="Homep
       if
                                not
self.can_edit_warehouse:
                                         стеллажа:").grid(row=row, column=0,
                                         padx=5, pady=5, sticky=W)
messagebox.showerror("Ошибка",
                                                    rack_var = StringVar()
вас нет прав на добавление деталей")
                                                    rack combobox
           return
                                         ttk.Combobox(add window,
                                         textvariable=rack var,
       try:
           add window
                                        values=[], state="readonly")
Toplevel(self.root)
                                         rack combobox.grid(row=row,
add window.title("Добавить деталь на
                                         column=1, padx=5, pady=5, sticky=EW)
                                                    row += 1
склад")
                                                    Label(add window,
self.cursor.execute("SELECT
                                         text="Homep полки:").grid(row=row,
warehouse id, warehouse number FROM
                                         column=0, padx=5, pady=5, sticky=W)
warehouse
                 ORDER BY
                                                    shelf_var = StringVar()
warehouse number")
                                                     shelf combobox
          warehouses
                                         ttk.Combobox(add window,
self.cursor.fetchall()
                                         textvariable=shelf var,
                                         values=[], state="readonly")
```

Изм.

Лист

№ докум.

Подпись Дата

KP.350000.000

```
shelf combobox.grid(row=row,
                                          room var.set('')
column=1, padx=5, pady=5, sticky=EW)
            row += 1
                                          rack var.set('')
            Label (add window,
                                          shelf var.set('')
text="Тип
            детали:").grid(row=row,
column=0, padx=5, pady=5, sticky=W)
                                          rack combobox['values'] = []
            type_entry
Entry(add window)
                                           shelf combobox['values'] = []
            type entry.grid(row=row,
                                                               except Exception
column=1, padx=5, pady=5, sticky=EW)
                                           as e:
            row += 1
                                          print(f"Error updating rooms: {e}")
            Label (add window,
              (кг):").grid(row=row,
text="Bec
                                           room combobox['values'] = []
column=0, padx=5, pady=5, sticky=W)
                                                           else:
            weight entry
Entry(add window)
                                           room combobox['values'] = []
weight entry.grid(row=row, column=1,
                                          rack combobox['values'] = []
padx=5, pady=5, sticky=EW)
            weight entry.insert(0,
                                          shelf combobox['values'] = []
"0.0")
            row += 1
                                                       def
                                          update racks(event=None):
            def
                                                           selected_warehouse =
update rooms (event=None):
                                          warehouse_var.get()
                selected warehouse =
                                                           selected room
warehouse var.get()
                                           room var.get()
                                                           if
                i f
                                           selected warehouse in warehouse_ids
selected warehouse in warehouse ids:
                    warehouse id =
                                           and selected room:
warehouse ids[selected warehouse]
                                                               warehouse id
                                           warehouse ids[selected warehouse]
                    try:
                                                               try:
self.cursor.execute("""
                                          self.cursor.execute("""
                            SELECT
                                                                       SELECT
room id, room number
                                          rack id, rack number
                            FROM
room
                                                                       FROM
                            WHERE
                                           rack
warehouseID = %s
                                                                       WHERE
                            ORDER BY
                                           roomID = (
room number
                        """,
                                           SELECT room id FROM room
(warehouse id,))
                                                   room number
                                                                            AND
                        rooms
self.cursor.fetchall()
                                           warehouseID = %s
                                                                       ORDER BY
room combobox['values']
[str(number) for id,
                        number
                                 in
                                          rack number
rooms]
                                           (int(selected room), warehouse id))
                        if rooms:
                                                                   racks
                                          self.cursor.fetchall()
room combobox.current(0)
                                          rack combobox['values']
room var.set(str(rooms[0][1]))
                                          [str(number) for id, number in
                                          racksl
update racks()
                        else:
                                                                   if racks:
```

Изм.	Лист	№ докум.	Подпись	Дата

```
""",
                                          (int(selected rack),
rack combobox.current(0)
                                          int(selected room), warehouse id))
rack var.set(str(racks[0][1]))
                                                                  shelves
                                          self.cursor.fetchall()
update shelves()
                                          shelf combobox['values']
                        else:
                                          [str(number) for id, number in
rack_var.set('')
                                          shelves]
                                                                  if shelves:
shelf var.set('')
shelf combobox['values'] = []
                                          shelf combobox.current(0)
                    except Exception
as e:
                                          shelf var.set(str(shelves[0][1]))
print(f"Error updating racks: {e}")
                                          shelf var.set('')
rack combobox['values'] = []
                                                              except Exception
                                          as e:
rack combobox['values'] = []
                                          print(f"Error updating shelves:
                                          {e}")
shelf combobox['values'] = []
                                          shelf combobox['values'] = []
            def
                                                          else:
update shelves(event=None):
                selected warehouse =
                                          shelf combobox['values'] = []
warehouse var.get()
                selected room
                                          warehouse combobox.bind("<<ComboboxS</pre>
room var.get()
                                          elected>>", update rooms)
               selected rack
rack var.get()
                                          room combobox.bind("<<ComboboxSelect</pre>
                if
(selected_warehouse in warehouse ids
                                          ed>>", update racks)
          selected room
                                          rack combobox.bind("<<ComboboxSelect</pre>
selected_rack):
                                          ed>>", update shelves)
                    warehouse id =
warehouse ids[selected warehouse]
                                                      if warehouse options:
self.cursor.execute("""
                                          warehouse combobox.current(0)
                            SELECT
shelf id, shelf number
                                          warehouse var.set(warehouse options[
                            FROM
shelf
                                                          update rooms()
                            WHERE
rackID = (
                                                      def save item():
                                                          try:
SELECT rack id FROM rack
                                                                           not
                                          all([warehouse_var.get(),
WHERE rack number = %s AND roomID =
                                          room var.get(),
                                          rack_var.get(), shelf_var.get(),
SELECT room id FROM room
                                          type entry.get(),
WHERE
      room_number = %s AND
                                          weight entry.get()]):
warehouseID = %s
                                                                  raise
                                          ValueError("Все поля должны быть
                                          заполнены")
                            )
                            ORDER BY
shelf number
                                          self.cursor.execute("""
```

```
SELECT
                                                                 raise
                                         ValueError(f"Ha полке
shelf id FROM shelf
                                                                          vже
                       WHERE
                                         {details count} деталей. Максимум -
shelf number = %s AND rackID = (
                                         5.")
                           SELECT
rack id FROM rack
                                         self.cursor.execute("""
                           WHERE
rack number = %s AND roomID = (
                                                                 INSERT INTO
                                         details (shelfid,
                                                                type_detail,
SELECT room id FROM room
                                         weight)
                                                                 VALUES (%s,
WHERE
        room number = %s AND
                                         %s, %s)
                                                             """, (shelf_id,
warehouseID = (
                                         type entry.get(), weight))
SELECT warehouse id FROM warehouse
WHERE warehouse number = %s
                                         self.conn.commit()
                                         self.load warehouse()
                                         add window.destroy()
(int(shelf var.get()),
int(rack var.get()),
                                         messagebox.showinfo("Успех", "Деталь
int(room var.get()),
                                         успешно добавлена")
int(warehouse var.get())))
                                                         except ValueError as
                                         ve:
                   shelf data
self.cursor.fetchone()
                                         messagebox.showerror("Ошибка",
                                         f"Неверные данные: {str(ve)}")
                                not
shelf data:
                                                         except Exception as
                       raise
ValueError("Полка не найдена в базе
данных")
                                         self.conn.rollback()
                   shelf id
shelf data[0]
                                         messagebox.showerror("Ошибка", f"He
                                         удалось добавить деталь: {str(e)}")
                   try:
                       weight
                                                     Button (add window,
float(weight entry.get())
                                         text="Сохранить",
                       if weight <=
                                         command=save item).grid(
0.0:
                                                         row=row,
                                                                    column=0,
                           raise
                                         columnspan=2, pady=10)
ValueError("Bec
                 не
                       может быть
отрицательным")
                                                 except Exception as e:
                   except
ValueError as e:
                                         messagebox.showerror("Ошибка", f"He
                                         удалось открыть форму: {str(e)}")
                       raise
ValueError("Вес должен быть числом
(например, 2.1)")
                                             def edit warehouse item(self):
                                                 """Редактирование детали на
                                         складе с динамическими выпадающими
                                         списками"""
self.cursor.execute("""
                       SELECT
                                                 if
                                                                          not
COUNT(*) FROM details WHERE shelfid
                                         self.can edit warehouse:
= %s
                                         messagebox.showerror("Ошибка",
(shelf id,))
                                         вас нет прав на редактирование
                   details count =
                                         деталей")
self.cursor.fetchone()[0]
                                                     return
                   if details count
                                                 selected
>= 5:
                                         self.warehouse tree.selection()
                                                 if not selected:
```

Подпись Дата

Изм.

Лист

№ докум.

```
warehouse_ids
messagebox.showwarning("Предупрежден {str(number): id for id, number in
ие", "Выберите
                    деталь для
                                        warehouses}
редактирования")
           return
                                                     row = 0
        item
                                                     Label (edit window,
                                         text="Homep склада:").grid(row=row,
self.warehouse tree.item(selected[0]
                                         column=0, padx=5, pady=5, sticky=W)
        detail id
                                                     warehouse var
item['values'][0]
                                         StringVar(value=str(detail data[10])
        try:
                                                     warehouse combobox
            self.cursor.execute("""
                                         ttk.Combobox(edit window,
                                         textvariable=warehouse var,
               SELECT
                   d.detail_id,
d.type detail, d.weight,
                                         values=warehouse options,
                   s.shelf id,
                                         state="readonly")
s.shelf number,
                   rk.rack id,
                                         warehouse combobox.grid(row=row,
rk.rack number,
                                         column=1, padx=5, pady=5, sticky=EW)
                   r.room id,
                                                    row += 1
r.room number,
                   w.warehouse id,
                                                    Label (edit window,
                                         text="Homep комнаты:").grid(row=row,
w.warehouse number
                                         column=0, padx=5, pady=5, sticky=W)
               FROM details d
               JOIN shelf s ON
                                                    room var
d.shelfid = s.shelf id
                                         StringVar(value=str(detail_data[8]))
               JOIN rack rk ON
                                                    room combobox
s.rackid = rk.rack id
                                         ttk.Combobox(edit window,
               JOIN
                     room r ON
                                         textvariable=room var,
rk.roomid = r.room id
               JOIN warehouse w ON
                                         values=[], state="readonly")
r.warehouseid = w.warehouse id
               WHERE d.detail_id =
                                         room combobox.grid(row=row,
                                         column=1, padx=5, pady=5, sticky=EW)
            """, (detail id,))
                                                     row += 1
           detail data
                                                     Label (edit window,
self.cursor.fetchone()
                                         text="Номер
                                         стеллажа:").grid(row=row, column=0,
           if not detail data:
                                         padx=5, pady=5, sticky=W)
                                                    rack var
messagebox.showerror("Ошибка",
                                         StringVar(value=str(detail data[6]))
"Деталь не найдена")
                                                    rack combobox
               return
                                         ttk.Combobox(edit window,
                                         textvariable=rack var,
           edit window
Toplevel(self.root)
                                         values=[], state="readonly")
edit window.title("Редактировать
                                         rack combobox.grid(row=row,
деталь")
                                         column=1, padx=5, pady=5, sticky=EW)
                                                     row += 1
self.cursor.execute("SELECT
                                                    Label(edit window,
warehouse id, warehouse number FROM
                                         text="Homep полки:").grid(row=row,
warehouse
                  ORDER BY
                                         column=0, padx=5, pady=5, sticky=W)
warehouse number")
                                                     shelf var
                                         StringVar(value=str(detail_data[4]))
           warehouses
self.cursor.fetchall()
                                                     shelf combobox
           warehouse options =
                                         ttk.Combobox(edit window,
[str(number) for id, number in
                                         textvariable=shelf var,
warehouses]
```

Изм.

Лист

№ докум.

Подпись Дата

KP.350000.000

Лист **О 1**

```
values=[], state="readonly")
                                          current room in [str(number) for id,
                                         number in rooms]:
shelf combobox.grid(row=row,
column=1, padx=5, pady=5, sticky=EW)
                                         room var.set(current room)
           row += 1
                                                                  elif rooms:
           Label(edit_window,
                                          room combobox.current(0)
            детали:").grid(row=row,
text="Тип
column=0, padx=5, pady=5, sticky=W)
                                          room var.set(str(rooms[0][1]))
           type var
StringVar(value=detail data[1])
                                          update racks()
            type entry
                                                              except Exception
Entry(edit_window,
                                          as e:
textvariable=type var)
           type entry.grid(row=row,
                                          print(f"Error updating rooms: {e}")
column=1, padx=5, pady=5, sticky=EW)
           row += 1
                                          room combobox['values'] = []
                                                          else:
           Label (edit window,
text="Bec
              (кг):").grid(row=row,
                                          room combobox['values'] = []
column=0, padx=5, pady=5, sticky=W)
           weight var
                                                      def
StringVar(value=str(detail data[2]))
                                          update racks(event=None):
           weight entry
                                                          selected warehouse =
Entry(edit window,
                                          warehouse var.get()
textvariable=weight var)
                                                          selected room
                                          room var.get()
weight_entry.grid(row=row, column=1,
                                                          if
padx=5, pady=5, sticky=EW)
                                          selected warehouse in warehouse ids
           row += 1
                                          and selected room:
                                                              warehouse id
                                          warehouse ids[selected warehouse]
            def
update_rooms(event=None):
                                                              try:
                selected warehouse =
                                          self.cursor.execute("""
warehouse_var.get()
                                                                      SELECT
                if
                                          rack id, rack number
selected warehouse in warehouse ids:
                    warehouse id =
                                                                      FROM
warehouse ids[selected warehouse]
                                          rack
                                                                      WHERE
                    try:
                                          roomID = (
self.cursor.execute("""
                            SELECT
                                          SELECT room id FROM room
room id, room number
                            FROM
                                          WHERE
                                                  room number
                                                                           AND
                                          warehouseID = %s
room
                            WHERE
warehouseID = %s
                                                                      ORDER BY
                            ORDER BY
                                          rack number
room number
                        """,
                                          (int(selected room), warehouse id))
(warehouse id,))
                                                                  racks
                        rooms
                                          self.cursor.fetchall()
self.cursor.fetchall()
                                          rack combobox['values']
room combobox['values']
                                          [str(number) for id, number in
[str(number) for id,
                        number in
                                          racksl
roomsl
                                                                  current rack
                                        = str(detail data[6])
                       current room
= str(detail data[8])
```

Изм.	Лист	№ докум.	Подпись	Дата

```
shelves
current rack in [str(number) for id, self.cursor.fetchall()
number in racks]:
                                         shelf combobox['values']
rack var.set(current rack)
                                         [str(number) for id, number
                                                                           in
                       elif racks:
                                         shelvesl
rack combobox.current(0)
                                         current shelf = str(detail data[4])
                                                                 if
rack var.set(str(racks[0][1]))
                                         current shelf in [str(number) for
                                         id, number in shelves]:
update shelves()
                   except Exception
as e:
                                         shelf var.set(current shelf)
print(f"Error updating racks: {e}")
                                         shelves:
rack combobox['values'] = []
                                         shelf combobox.current(0)
               else:
                                         shelf var.set(str(shelves[0][1]))
rack combobox['values'] = []
                                                             except Exception
                                         as e:
           def
                                         print(f"Error updating shelves:
update shelves(event=None):
               selected warehouse =
                                         {e}")
warehouse var.get()
                                         shelf combobox['values'] = []
               selected room
room var.get()
                                                         else:
               selected rack
rack var.get()
                                         shelf combobox['values'] = []
               if
(selected_warehouse in warehouse ids
                                         warehouse combobox.bind("<<ComboboxS</pre>
          selected room and
                                         elected>>", update rooms)
selected rack):
                   warehouse_id =
warehouse_ids[selected_warehouse]
                                         room combobox.bind("<<ComboboxSelect</pre>
                                         ed>>", update_racks)
                   try:
self.cursor.execute("""
                                         rack combobox.bind("<<ComboboxSelect</pre>
                           SELECT
                                         ed>>", update shelves)
shelf id, shelf number
                           FROM
                                                     update rooms()
shelf
                           WHERE
                                                     def save changes():
rackID = (
                                                         try:
                                                                          not
SELECT rack id FROM rack
                                         all([warehouse var.get(),
                                         room var.get(),
WHERE rack number = %s AND roomID =
                                         rack_var.get(), shelf_var.get(),
SELECT room id FROM room
                                         type_var.get(), weight_var.get()]):
                                                                 raise
WHERE room number
                         %s
                              AND
                                         ValueError("Все поля должны быть
warehouseID = %s
                                         заполнены")
                               )
                           ORDER BY
                                         self.cursor.execute("""
shelf number
                                                                 SELECT
                                         shelf id FROM shelf
(int(selected rack),
                                                                 WHERE
int(selected room), warehouse id))
                                         shelf number = %s AND rackID = (
```

```
SELECT
                                         {details count} деталей. Максимум -
rack id FROM rack
                                         5.")
                           WHERE
rack number = %s AND roomID = (
                                         self.cursor.execute("""
SELECT room id FROM room
                                                                 UPDATE
                                         details
        room number = %s AND
                                                                 SET shelfid
WHERE
warehouseID = (
                                         = %s, type detail = %s, weight = %s
SELECT warehouse id FROM warehouse
                                         detail id = %s
                                                             """,
WHERE warehouse number = %s
                                         (new shelf id,
                                                             type var.get(),
                                         weight, detail id))
                               )
                                         self.conn.commit()
(int(shelf var.get()),
int(rack var.get()),
                                         self.load warehouse()
int(room var.get()),
int(warehouse var.get())))
                                         edit window.destroy()
                   shelf data
                                         messagebox.showinfo("Успех", "Деталь
self.cursor.fetchone()
                                         успешно обновлена")
                                                         except ValueError as
                                not
shelf data:
                                         ve:
ValueError("Полка не найдена в базе
                                         messagebox.showerror("Ошибка",
данных")
                                         f"Неверные данные: {str(ve)}")
                   new shelf id =
                                                        except Exception as
shelf data[0]
                                         е:
                                         self.conn.rollback()
                   try:
                       weight
float(weight var.get())
                                         messagebox.showerror("Ошибка", f"He
                       if weight <=
                                         удалось обновить деталь: {str(e)}")
0.0:
                                                     Button(edit_window,
                           raise
ValueError("Bec
                                         text="Сохранить",
                       может быть
                не
отрицательным")
                                         command=save changes).grid(
                   except
                                                         row=row, column=0,
ValueError as e:
                                         columnspan=2, pady=10)
                       raise
ValueError("Вес должен быть числом
                                                 except Exception as e:
(например, 2.1)")
                                         messagebox.showerror("Ошибка", f"He
                   if new shelf id
                                         удалось открыть
                                                                        форму
!= detail data[3]:
                                         редактирования: {str(e)}")
self.cursor.execute("""
                                             def delete warehouse item(self):
                                                 """Удаление детали со
                           SELECT
                                         склада"""
COUNT(*) FROM details WHERE shelfid
= %s
                                                 selected
                       """,
                                         self.warehouse_tree.selection()
                                                 if not selected:
(new shelf id,))
details count
                                         messagebox.showwarning("Предупрежден
self.cursor.fetchone()[0]
                                         ие", "Выберите деталь для удаления")
                                                     return
                       if
details count >= 5:
                                                 item
                           raise
                                         self.warehouse tree.item(selected[0]
ValueError(f"На новой полке уже
```

```
detail id
                                         вас нет
                                                     прав на добавление
item['values'][0]
                                         накладных")
                                                    return
       self.cursor.execute("SELECT
                                                 trv:
COUNT(*) FROM invoice detail WHERE
                                                     add window
detailid = %s", (detail_id,))
                                         Toplevel (self.root)
       reference count
self.cursor.fetchone()[0]
                                         add window.title("Добавить
                                         контрагента")
       if reference count > 0:
                                                    Label (add window,
                                         text="Hазвание:").grid(row=0,
messagebox.showerror("Ошибка",
                                         column=0, padx=5, pady=5, sticky=W)
f"Невозможно удалить деталь: она
                                                    name entry
                                         Entry(add window)
используется в {reference count}
накладных")
                                                    name entry.grid(row=0,
           return
                                         column=1, padx=5, pady=5, sticky=EW)
       if
                                                    Label(add window,
messagebox.askyesno("Подтверждение",
                                         text="Контактное лицо:").grid(row=1,
f"Вы уверены, что хотите удалить
                                         column=0, padx=5, pady=5, sticky=W)
деталь №{detail id}?"):
                                                     contact entry
                                         Entry(add window)
           try:
self.cursor.execute("DELETE
                               FROM
                                         contact entry.grid(row=1, column=1,
                                         padx=5, pady=5, sticky=EW)
details WHERE detail id = %s",
(detail id,))
               self.conn.commit()
                                                    Label(add window,
                                         text="Телефон:").grid(row=2,
self.load warehouse()
                                         column=0, padx=5, pady=5, sticky=W)
                                                    phone entry
               if
                                         Entry(add window)
                                                    phone entry.grid(row=2,
len(self.warehouse tree.get children
                   0
                                         column=1, padx=5, pady=5, sticky=EW)
self.current_search_conditions['ware
house']:
                                                    Label (add window,
                                         text="Aдpec:").grid(row=3, column=0,
                                         padx=5, pady=5, sticky=W)
messagebox.showinfo("Информация",
"Таблица
         пуста. Сброс условий
                                                     address entry
поиска.")
                                         Entry(add window)
self.current search conditions['ware
                                         address entry.grid(row=3, column=1,
house'] = None
                                         padx=5, pady=5, sticky=EW)
self.load warehouse()
                                                     def save counteragent():
                                        self.cursor.execute("""
messagebox.showinfo("Успех", "Деталь
успешно удалена")
                                                                INSERT INTO
           except Exception as e:
                                         counteragent (counteragent name,
               self.conn.rollback()
                                         contact_person, phone_number,
                                         address)
messagebox.showerror("Ошибка", f"He
                                                                VALUES (%s,
удалось удалить деталь: {str(e)}")
                                         %s, %s, %s)
                                                            """, (
   def add counteragent(self):
       """Добавление нового
                                        name entry.get(),
контрагента"""
       if
                                         contact entry.get(),
                                not
self.can edit counteragents:
                                         int(phone entry.get()),
messagebox.showerror("Ошибка",
                                         address entry.get()
```

KP.350000.000

```
))
                                                      if
                                                                           not
                                          counteragent data:
self.conn.commit()
                                          messagebox.showerror("Ошибка",
                                          "Контрагент не найден")
self.load counteragents()
                                                          return
add window.destroy()
                                                      edit window
                                          Toplevel(self.root)
messagebox.showinfo("Успех",
"Контрагент успешно добавлен")
                                          edit window.title("Редактировать
                except Exception as
                                          контрагента")
e:
                                                      Label (edit window,
self.conn.rollback()
                                          text="Haзвание:").grid(row=0,
                                          column=0, padx=5, pady=5, sticky=W)
messagebox.showerror("Ошибка", f"He
                                                      name entry
          добавить контрагента:
                                          Entry(edit_window)
{str(e)}")
                                                      name entry.insert(0,
                                          counteragent data[1])
           Button (add window,
                                                      name entry.grid(row=0,
text="Сохранить",
                                          column=1, padx=5, pady=5, sticky=EW)
command=save counteragent).grid(row=
4, column=0, columnspan=2, pady=10)
                                                      Label(edit window,
                                          text="Контактное лицо: ").grid(row=1,
        except Exception as e:
                                          column=0, padx=5, pady=5, sticky=W)
                                                      contact entry
messagebox.showerror("Ошибка", f"He
                                          Entry(edit_window)
удалось открыть форму: {str(e)}")
                                                      contact entry.insert(0,
                                          counteragent data[2])
    def edit counteragent(self):
        """Редактирование
                                          contact entry.grid(row=1, column=1,
контрагента"""
                                          padx=5, pady=5, sticky=EW)
        i f
                                 not.
                                                      Label (edit window,
self.can_edit_counteragents:
                                          text="Телефон:").grid(row=2,
messagebox.showerror("Ошибка",
                                          column=0, padx=5, pady=5, sticky=W)
                                                      phone entry
вас нет
           прав на добавление
накладных")
                                          Entry(edit window)
                                                      phone entry.insert(0,
            return
                                          str(counteragent data[3]))
       selected
self.counteragent tree.selection()
                                                      phone entry.grid(row=2,
       if not selected:
                                          column=1, padx=5, pady=5, sticky=EW)
messagebox.showwarning("Предупрежден
                                                      Label(edit window,
ие", "Выберите
                 контрагента для
                                          text="Aдpec:").grid(row=3, column=0,
редактирования")
                                          padx=5, pady=5, sticky=W)
                                                      address entry
           return
                                          Entry(edit window)
                                                      address entry.insert(0,
self.counteragent tree.item(selected
                                          counteragent data[4])
        counteragent id
                                          address entry.grid(row=3, column=1,
                                          padx=5, pady=5, sticky=EW)
item['values'][0]
       try:
                                                      def save_changes():
                                                          try:
self.cursor.execute("SELECT * FROM
counteragent WHERE counteragent id =
                                         self.cursor.execute("""
%s", (counteragent id,))
                                                                  UPDATE
            counteragent data =
                                         counteragent
self.cursor.fetchone()
                                                                  SET
```

Изм.	Лист	№ докум.	Подпись	Дата

```
item
counteragent name = %s,
                                        self.counteragent tree.item(selected
contact person = %s,
                                                counteragent id
                                        item['values'][0]
phone number = %s,
                                                counteragent name
                          address
                                        item['values'][1]
= %s
                     WHERE
                                        messagebox.askyesno("Подтверждение",
counteragent_id = %s
                                        f"Вы уверены, что хотите удалить
                                        контрагента
                                         '{counteragent name}'?"):
name entry.get(),
                                                    try:
contact entry.get(),
                                        self.cursor.execute("DELETE FROM
int(phone entry.get()),
                                        counteragent WHERE counteragent id =
                                        %s", (counteragent id,))
address_entry.get(),
                                                        self.conn.commit()
counteragent id
                                        self.load counteragents()
                   ))
                                        len(self.counteragent_tree.get_child
self.conn.commit()
                                        self.current_search_conditions['coun
self.load counteragents()
                                        teragent']:
edit window.destroy()
                                        messagebox.showinfo("Информация",
                                        "Таблица пуста. Сброс условий
messagebox.showinfo("Успех",
                                        поиска.")
"Контрагент успешно обновлен")
               except Exception as
                                        self.current search conditions['coun
                                        teragent'] = None
self.conn.rollback()
                                        self.load counteragents()
messagebox.showerror("Ошибка", f"He
удалось обновить контрагента:
                                        messagebox.showinfo("Успех",
{str(e)}")
                                        "Контрагент успешно удален")
                                                    except Exception as e:
           Button(edit window,
                                                        self.conn.rollback()
text="Сохранить",
command=save_changes).grid(row=4,
                                        messagebox.showerror("Ошибка", f"He
column=0, columnspan=2, pady=10)
                                        удалось удалить контрагента:
                                        {str(e)}")
       except Exception as e:
                                            def add employee(self):
                                             """\overline{\mathbb{Q}}обавление
messagebox.showerror("Ошибка", f"He
удалось открыть форму: {str(e)}")
                                        сотрудника"""
                                                                        not
   def delete_counteragent(self):
                                        self.can edit employees:
       """Удаление контрагента"""
                                        messagebox.showerror("Ошибка",
       selected
self.counteragent tree.selection()
                                        вас нет
                                                   прав на добавление
       if not selected:
                                        накладных")
                                                    return
messagebox.showwarning("Предупрежден
                                                try:
ие", "Выберите контрагента для
                                                    add window
удаления")
                                        Toplevel(self.root)
           return
                                        add window.title("Добавить
                                         сотрудника")
```

```
Label (add window,
                                         self.conn.commit()
text="Роль:").grid(row=0, column=0,
padx=5, pady=5, sticky=W)
                                         self.load employees()
           role var = StringVar()
           role combobox
                                         add window.destroy()
ttk.Combobox(add window,
textvariable=role var,
                                         messagebox.showinfo("Усπех",
                                         "Сотрудник успешно добавлен")
values=["Кладовщик",
                          "Менеджер
                                                        except Exception as
склада", "Владелец"])
                                         self.conn.rollback()
role combobox.grid(row=0, column=1,
padx=5, pady=5, sticky=EW)
                                         messagebox.showerror("Ошибка", f"He
           Label(add window,
                                                    добавить сотрудника:
                                         удалось
                                         {str(e)}")
text="Фамилия:").grid(row=1,
column=0, padx=5, pady=5, sticky=W)
           last name entry
                                                    Button(add window,
Entry(add window)
                                         text="Сохранить",
                                         command=save employee).grid(row=4,
last name entry.grid(row=1,
                                         column=0, columnspan=2, pady=10)
column=1, padx=5, pady=5, sticky=EW)
                                                 except Exception as e:
           Label(add window,
text="Mms:").grid(row=2, column=0,
                                         messagebox.showerror("Ошибка", f"He
padx=5, pady=5, sticky=W)
                                         удалось открыть форму: {str(e)}")
           first name entry
Entry(add window)
                                             def edit employee(self):
                                                 """Редактирование
first name_entry.grid(row=2,
                                         сотрудника"""
column=1, padx=5, pady=5, sticky=EW)
                                                                          not
                                         self.can edit employees:
           Label (add window,
                                         messagebox.showerror("Ошибка",
text="Отчество:").grid(row=3,
column=0, padx=5, pady=5, sticky=W)
                                         вас нет прав на добавление
                                         накладных")
           patronymic entry
Entry(add window)
                                                     return
                                                 selected
                                         self.employee tree.selection()
patronymic entry.grid(row=3,
column=1, padx=5, pady=5, sticky=EW)
                                                 if not selected:
            def save_employee():
                                         messagebox.showwarning("Предупрежден
                                         ие", "Выберите сотрудника для
                                         редактирования")
self.cursor.execute("""
                       INSERT INTO
employee (employee role, last name,
first name, patronymic)
                                         self.employee tree.item(selected[0])
                       VALUES (%s,
                                                employee id
                                         item['values'][0]
%s, %s, %s)
                   """, (
                                                 try:
role var.get(),
                                         self.cursor.execute("SELECT * FROM
last name entry.get(),
                                         employee WHERE employee id = %s",
                                         (employee id,))
first name entry.get(),
                                                     employee data
                                         self.cursor.fetchone()
patronymic entry.get()
                                                     if not employee data:
```

Изм.	Лист	№ докум.	Подпись	Дата

```
def save changes():
messagebox.showerror("Ошибка",
                                                         try:
"Сотрудник не найден")
                                        self.cursor.execute("""
               return
                                                                 UPDATE
           edit window
                                         employee
Toplevel(self.root)
                                                                 SET
edit window.title("Редактировать
                                        employee role = %s,
сотрудника")
                                        last name = %s,
           Label(edit window,
text="Роль:").grid(row=0, column=0,
                                        first name = %s,
padx=5, pady=5, sticky=W)
           role var = StringVar()
                                        patronymic = %s
           role combobox
                                                                 WHERE
ttk.Combobox(edit window,
                                         employee id = %s
                                                             """, (
textvariable=role var,
values=["Кладовщик",
                          "Менеджер
                                         role var.get(),
склада", "Владелец"])
                                         last name entry.get(),
role combobox.set(employee data[1])
                                         first name entry.get(),
role combobox.grid(row=0, column=1,
padx=5, pady=5, sticky=EW)
                                         patronymic entry.get(),
                                                                 employee id
           Label (edit window,
                                                             ))
text="Фамилия:").grid(row=1,
column=0, padx=5, pady=5, sticky=W)
           last_name_entry
                                        self.conn.commit()
Entry(edit window)
                                         self.load employees()
last name entry.insert(0,
employee_data[2])
                                         edit window.destroy()
                                         messagebox.showinfo("Успех",
last_name_entry.grid(row=1,
column=1, padx=5, pady=5, sticky=EW)
                                         "Сотрудник успешно обновлен")
                                                        except Exception as
           Label(edit window,
                                         e:
text="Имя:").grid(row=2, column=0,
padx=5, pady=5, sticky=W)
                                         self.conn.rollback()
           first_name_entry =
Entry(edit window)
                                         messagebox.showerror("Ошибка", f"He
                                         удалось
                                                 обновить сотрудника:
first name entry.insert(0,
                                         {str(e)}")
employee data[3])
                                                    Button(edit window,
first name entry.grid(row=2,
                                         text="Сохранить",
column=1, padx=5, pady=5, sticky=EW)
                                         command=save changes).grid(row=4,
                                         column=0, columnspan=2, pady=10)
           Label (edit window,
text="Отчество:").grid(row=3,
                                                except Exception as e:
column=0, padx=5, pady=5, sticky=W)
                                        messagebox.showerror("Ошибка", f"He
           patronymic_entry =
Entry(edit window)
                                        удалось открыть форму: {str(e)}")
patronymic entry.insert(0,
                                             def delete employee(self):
                                                 """Удаление сотрудника"""
employee data[4])
                                                 selected
patronymic entry.grid(row=3,
                                        self.employee tree.selection()
column=1, padx=5, pady=5, sticky=EW)
                                                 if not selected:
```

Изм.	Лист	№ докум.	Подпись	Дата

```
messagebox.showwarning("Предупрежден
ие", "Выберите сотрудника для
                                       len(self.employee_tree.get_children(
удаления")
                                                 ==
                                                         0
                                                                       and
                                        ) )
           return
                                        self.current search conditions['empl
                                        oyee']:
       item
                                        messagebox.showinfo("Информация",
self.employee_tree.item(selected[0])
                                        "Таблица пуста. Сброс условий
       employee_id
item['values'][0]
                                        поиска.")
       employee name
f"{item['values'][2]}
                                        self.current search conditions['empl
{item['values'][3]}
                                        oyee'] = None
{item['values'][4]}"
                                        self.load employees()
       if
messagebox.askyesno("Подтверждение",
f"Вы уверены, что хотите удалить
                                        messagebox.showinfo("Успех",
coтрудника '{employee_name}'?"):
                                        "Сотрудник успешно удален")
                                                  except Exception as e:
           try:
                                                       self.conn.rollback()
self.cursor.execute("DELETE FROM
employee WHERE employee id = %s",
                                       messagebox.showerror("Ошибка", f"He
(employee_id,))
                                       удалось удалить сотрудника:
               self.conn.commit()
                                       {str(e)}")
self.load employees()
                                        WindowApp().auth window()
```

	·			
Изм.	Лист	№ докум.	Подпись	Дата

Приложение Д Исходный код базы данных склада запчастей

Листинг 1 – Исходный код базы данных склада запчастей

```
SET statement timeout = 0;
                                             DELETE FROM invoice detail WHERE
SET lock timeout = 0;
                                         invoiceID = OLD.invoice id;
                                            DELETE
                                                    FROM
                                                            invoice
                                         invoice id = OLD.invoice id;
idle in transaction session timeout
                                            RETURN OLD;
SET transaction timeout = 0;
                                         END:
SET client encoding = 'UTF8';
                                         $$;
    standard conforming strings =
                                        ALTER
                                                                    FUNCTION
on;
SELECT
                                        public.delete_invoice_details_view()
                                        OWNER TO postgres;
pg_catalog.set_config('search_path',
'', false);
SET check function bodies = false;
                                        CREATE
                                                                    FUNCTION
SET xmloption = content;
                                        public.delete related data() RETURNS
SET client min messages = warning;
                                         trigger
SET row security = off;
                                            LANGUAGE
                                                                  SECURITY
                                                       plpgsql
                                         DEFINER
CREATE
                           FUNCTION
                                            AS $$
public.convert_text_to_boolean(text_
                                         BEGIN
value text, field type text DEFAULT
                                            IF TG_TABLE NAME = 'warehouse'
'status'::text) RETURNS boolean
                                         THEN
   LANGUAGE plpgsql IMMUTABLE
                                                PERFORM * FROM details WHERE
   AS $$
                                         shelfID IN (
BEGIN
                                                    SELECT shelf id
                                                                        FROM
    text value
                                         shelf WHERE rackID IN (
LOWER(TRIM(text value));
                                                        SELECT rack id FROM
   IF field type = 'type' THEN
                                        rack WHERE roomID IN (
       RETURN text_value
                                                            SELECT room id
                       'выгрузить',
                                        FROM room WHERE
('выгрузка',
                                                            warehouseID =
'отправка',
             'true',
                     '1', 'да',
                                        OLD.warehouse_id
'yes', 'y');
   ELSE
                                                    )
                                                ) LIMIT 1;
       RETURN
                 text value
('завершено', 'готово', 'выполнено',
                                                DELETE FROM shelf WHERE
'done', 'true', '1', 'да', 'yes',
                                        rackID IN (
'y');
                                                    SELECT rack_id FROM rack
   END IF;
                                         WHERE roomID IN (
                                                        SELECT room id FROM
END:
                                                 WHERE
$$;
                                                          warehouseID
                                         room
                                         OLD.warehouse_id
                           FUNCTION
                                                    )
public.convert text to boolean(text
                                                );
                                                DELETE
value text, field type text) OWNER
                                                        FROM rack WHERE
TO postgres;
                                         roomID IN (
                                                    SELECT room id FROM room
                           FUNCTION
                                         WHERE warehouseID = OLD.warehouse id
CREATE
public.delete_invoice_details_view()
                                                );
                                                DELETE FROM room
RETURNS trigger
                                                                     WHERE
               plpgsql
                          SECURITY
                                         warehouseID = OLD.warehouse id;
   LANGUAGE
                                            ELSIF TG TABLE_NAME = 'room'
DEFINER
   AS $$
                                         THEN
                                                PERFORM * FROM details WHERE
BEGIN
   DELETE FROM
                   invoice employee
                                         shelfID IN (
WHERE invoiceID = OLD.invoice id;
                                                    SELECT
                                                            shelf id
                                         shelf WHERE rackID IN (
```

Изм.	Лист	№ докум.	Подпись	Дата

```
FROM shelf s
WHERE s.rackID = rack_id_value
               SELECT rack id FROM
rack WHERE roomID = OLD.room id
                                                 s.shelf_number
                                       AND
          )
       ) LIMIT 1;
                                       OLD.shelf number;
       DELETE FROM shelf WHERE
                                          DELETE FROM
                                                         details WHERE
rackID IN (
                                       detail id = OLD.detail id;
                                         DELETE FROM shelf
           SELECT rack id FROM rack
WHERE roomID = OLD.room id
                                           WHERE shelf.shelf id
                                       shelf id value
      );
       DELETE FROM rack WHERE
                                            AND NOT EXISTS (SELECT 1 FROM
roomID = OLD.room id;
                                       details WHERE details.shelfID =
   ELSIF TG TABLE NAME = 'rack'
                                       shelf.shelf id);
                                          DELETE FROM rack
       PERFORM * FROM details WHERE
                                          WHERE rack.rack id
shelfID IN (
                                       rack id value
          SELECT shelf id FROM
                                           AND NOT EXISTS (SELECT 1 FROM
shelf WHERE rackID = OLD.rack_id
                                              WHERE shelf.rackID =
       ) LIMIT 1;
                                       rack.rack id);
       DELETE FROM shelf WHERE
                                          DELETE FROM room
rackID = OLD.rack id;
                                          WHERE room.room id
   ELSIF TG TABLE NAME = 'shelf'
                                       room id value
                                           AND NOT EXISTS (SELECT 1 FROM
      PERFORM * FROM details WHERE
                                               WHERE
                                       rack
                                                       rack.roomID
shelfID = OLD.shelf id LIMIT 1;
                                       room.room id);
   END IF;
                                          DELETE FROM warehouse
                                           WHERE warehouse.warehouse_id =
   RETURN OLD;
END:
                                       warehouse id value
                                            AND NOT EXISTS (SELECT 1 FROM
$$;
                                       room WHERE room.warehouseID =
                         FUNCTION
                                       warehouse.warehouse id);
RETURN OLD;
                                       END;
TO postgres;
                                       $$;
                          FUNCTION
CREATE
public.delete_warehouse_details()
                                       ALTER
                                       public.delete_warehouse_details()
RETURNS trigger
                                       OWNER TO postgres;
   LANGUAGE plpgsql
   AS $$
                                       CREATE
DECLARE
                                       public.get_employee_id(p_last_name
   warehouse id value integer;
                                       character varying, p_first_name character varying, p_patronymic
   room id value integer;
   rack id value integer;
   shelf_id_value integer;
                                       character varying) RETURNS integer
                                          LANGUAGE plpgsql
           w.warehouse_id INTO
   SELECT
                                       DEFINER
warehouse id value
                                          AS $$
   FROM warehouse w
                                       DECLARE
   WHERE w.warehouse_number
                                          v id integer;
                                       BEGIN
OLD.warehouse number;
   SELECT
             r.room id INTO
                                           SELECT employee id INTO v id
room id value
                                           FROM employee
   FROM room r
                                           WHERE last_name = p_last_name
          r.warehouseID =
   WHERE
                                          AND first_name = p_first_name
                                           AND patronymic = p_patronymic;
warehouse id value AND r.room number
= OLD.room number;
                                           IF NOT FOUND THEN
   SELECT
              ra.rack id INTO
                                              RAISE EXCEPTION 'Сотрудник %
rack id value
                                      % % не найден',
   FROM rack ra
                                                 p_last_name,
   WHERE ra.roomID = room id value
                                       p_first_name, p_patronymic;
         ra.rack_number =
                                         END IF;
OLD.rack number;
                                          RETURN v id;
              s.shelf id INTO
   SELECT
                                       END:
shelf id value
                                       $$;
```

Лист № докум. Подпись Дата

```
ELSE
ALTER
                          FUNCTION
                                             SELECT rack.rack id INTO
public.get employee id(p last name
                                       rack id FROM rack WHERE rack number
character varying, p_first_name
                                      = NEW.rack number AND roomID =
                     p_patronymic
character
           varying,
                                       room id;
character
          varying)
                      OWNER TO
                                          END IF;
                                          IF NOT EXISTS (SELECT 1 FROM
postgres;
                                       shelf WHERE shelf_number =
                          FUNCTION
                                       NEW.shelf number AND rackID
CREATE
                                       rack_id) THEN
public.insert_into_warehouse_details
() RETURNS trigger
                                              INSERT
                                                         INTO
                                                                   shelf
   LANGUAGE plpgsql
                                       (shelf_number, rackID)
                                              VALUES (NEW.shelf number,
   AS $$
DECLARE
                                       rack id)
                                              RETURNING
   warehouse id integer;
                                                          shelf.shelf id
   room id integer;
                                       INTO shelf id;
   rack id integer;
   shelf_id integer;
                                              SELECT shelf.shelf_id INTO
                                       shelf id FROM shelf WHERE
   IF NOT EXISTS (SELECT 1 FROM
                                       shelf number = NEW.shelf number AND
warehouse WHERE warehouse number =
                                       rackID = rack id;
NEW.warehouse number) THEN
                                          END IF;
       INSERT
              INTO warehouse
                                          INSERT INTO details (shelfID,
(warehouse number, address)
                                       weight, type detail)
       VALUES
                                          VALUES (shelf id, NEW.weight,
(NEW.warehouse_number, 'default
                                       NEW.type detail);
address')
                                          RETURN NEW;
      RETURNING
                                       END;
                                       $$;
warehouse.warehouse id
                            INTO
warehouse id;
   ELSE
                                       ALTER
       SELECT
                                       public.insert into warehouse details
warehouse.warehouse id
                            INTO
                                       () OWNER TO postgres;
warehouse id FROM warehouse WHERE
warehouse number
                                       CREATE
                                                                 FUNCTION
                                       public.insert_invoice_details_view()
NEW.warehouse number;
   END IF;
                                       RETURNS trigger
   IF NOT EXISTS (SELECT 1 FROM
                                          LANGUAGE plpgsql
       WHERE room number =
                                          AS $$
NEW.room number AND warehouseID =
                                       DECLARE
warehouse id) THEN
                                          v invoice id INTEGER;
       INSERT
                   INTO
                                          v counteragent id INTEGER;
(room number, warehouseID)
                                          v detail id INTEGER;
      VALUES (NEW.room number,
                                          v_employee_id INTEGER;
                                          v_type_invoice BOOLEAN;
warehouse id)
                                          v status BOOLEAN;
      RETURNING room.room id INTO
room id;
                                       BEGIN
   ELSE
                                          SELECT counteragent id INTO
       SELECT room.room id INTO
                                       v counteragent id
room id FROM room WHERE room number
                                          FROM counteragent
                                          WHERE counteragent_name
= NEW.room number AND warehouseID =
warehouse id;
                                       NEW.counteragent name;
   END IF;
                                          IF NOT FOUND THEN
   IF NOT EXISTS (SELECT 1 FROM
                                             RAISE EXCEPTION 'Контрагент
rack WHERE rack number =
                                       С
                                           именем % не найден',
NEW.rack number
                AND roomID
                                       NEW.counteragent_name;
room id) THEN
                                          END IF;
       INSERT
                  INTO rack
                                          IF NEW.type_invoice_text IS NOT
(rack number, roomID)
                                       NULL THEN
       VALUES (NEW.rack_number,
                                              v type invoice
                                       convert text to boolean (NEW.type inv
       RETURNING rack.rack id INTO
                                       oice text, 'type');
rack id;
                                          ELSE
```

Лист

№ докум.

Подпись Дата

KP.350000.000

```
ON CONFLICT (invoiceID,
       v type invoice
COALESCE (NEW.type invoice bool,
                                       detailID)
                                          DO UPDATE SET quantity =
FALSE);
   END IF;
                                        invoice detail.quantity
   IF NEW.status_text IS NOT NULL
                                       NEW.quantity;
THEN
                                           v employee id
                                       get employee id(
       v status
convert_text_to_boolean(NEW.status_t
                                               NEW.responsible_last_name,
ext, 'status');
                                               NEW.responsible first name,
   ELSE
                                               NEW.responsible_patronymic
       v status
COALESCE (NEW.status bool, FALSE);
                                           INSERT INTO invoice_employee
                                        (invoiceID,
   END IF;
                                                             responsible,
   IF NEW.invoice id IS NOT NULL
                                       granted_access, when_granted)
                                           VALUES (v invoice id,
       PERFORM 1 FROM invoice WHERE
                                       v_employee_id, v_employee_id, NOW())
invoice id = NEW.invoice id;
                                          ON CONFLICT
                                                              (invoiceID,
       IF FOUND THEN
                                       responsible) DO NOTHING;
           UPDATE invoice SET
                                          NEW.invoice_id := v_invoice_id;
                                          RETURN NEW;
               counteragentID
v_counteragent_id,
                                       END;
                                       $$;
              date time
NEW.date time,
               type invoice
                                       ALTER
                                                                  FUNCTION
                                       public.insert invoice details view()
v type invoice,
               status = v status
                                       OWNER TO postgres;
           WHERE invoice id
                                       CREATE
                                                                  FUNCTION
NEW.invoice id;
           v invoice id
                                       public.log counteragent changes()
                                       RETURNS trigger
NEW.invoice id;
                                           LANGUAGE plpgsql
       ELSE
           INSERT INTO invoice
                                           AS $$
(invoice id,
                  counteragentID,
                                       BEGIN
                                          IF (TG_OP = 'INSERT') THEN
date_time, type_invoice, status)
                                               INSERT
           VALUES (NEW.invoice id,
                                                        INTO
                                                                log_table
                                        (table_name, action_type, record_id,
v counteragent id,
                   NEW.date time,
v_type_invoice, v_status)
                                        new_values)
          RETURNING
                                              VALUES
                       invoice id
                                                         ('counteragent',
                                        'INSERT',
INTO v invoice id;
                                                      NEW.counteragent id,
       END IF;
                                        to jsonb(NEW));
   ELSE
                                              RETURN NEW;
                                           ELSIF (TG OP = 'UPDATE') THEN
       INSERT
                 INTO
                          invoice
                                                        INTO log table
(counteragentID,
                       date_time,
                                              INSERT
type invoice, status)
                                        (table_name, action_type, record_id,
       VALUES (v_counteragent_id,
                                        old_values, new_values)
NEW.date time,
                  v type invoice,
                                              VALUES
                                                        ('counteragent',
                                        'UPDATE',
                                                      OLD.counteragent id,
v status)
       RETURNING invoice id INTO
                                        to_jsonb(OLD), to_jsonb(NEW));
v invoice id;
                                               RETURN NEW;
   END IF;
                                           ELSIF (TG OP = 'DELETE') THEN
                                              INSERT INTO log table
   SELECT
              detail id INTO
v_detail_id
                                        (table_name, action_type, record_id,
   FROM details
                                        old values)
   WHERE
              type_detail
                                              VALUES
                                                         ('counteragent',
NEW.type detail;
                                        'DELETE',
                                                      OLD.counteragent id,
   IF NOT FOUND THEN
                                        to_jsonb(OLD));
       RAISE EXCEPTION 'Деталь типа
                                              RETURN OLD;
% не найдена', NEW.type detail;
                                           END IF;
   END IF;
                                       END;
   INSERT
            INTO
                                       $$;
                   invoice detail
(invoiceID, detailID, quantity)
                    (v invoice id,
v detail id, NEW.quantity)
```

```
FUNCTION
                                                  INSERT
                                                           INTO
ALTER
                                                                    log table
                                          (table_name, action_type, record_id,
public.log counteragent changes()
OWNER TO postgres;
                                          old values)
                                                  VALUES
                                                                  ('employee',
                            FUNCTION
                                          'DELETE',
                                                              OLD.employee id,
CREATE
public.log_details_changes() RETURNS
                                          to_jsonb(OLD));
                                                  RETURN OLD;
    LANGUAGE plpgsql
                                              END IF;
   AS $$
                                          END;
BEGIN
                                          $$;
    IF (TG OP = 'INSERT') THEN
        INSERT INTO log table
                                          ALTER
                                                                      FUNCTION
(table name, action type, record id,
                                          public.log employee changes() OWNER
new values)
                                          TO postgres;
        VALUES ('details', 'INSERT',
NEW.detail id, to jsonb(NEW));
                                          CREATE
                                                                      FUNCTION
       RETURN NEW;
                                          public.log_invoice_changes() RETURNS
    ELSIF (TG OP = 'UPDATE') THEN
       INSERT
                 INTO
                          log table
                                             LANGUAGE plpgsql
                                             AS $$
(table name, action type, record id,
                                          BEGIN
old values, new values)
       VALUES ('details', 'UPDATE',
                                              IF (TG OP = 'INSERT') THEN
                                                 INSERT
                                                            INTO
OLD.detail id,
                     to jsonb(OLD),
                                                                    log table
to_jsonb(NEW));
                                          (table_name, action_type, record_id,
                                          new_values)
       RETURN NEW;
    ELSIF (TG OP = 'DELETE') THEN
                                                  VALUES ('invoice', 'INSERT',
                 INTO
       INSERT
                         log_table
                                          NEW.invoice_id, to_jsonb(NEW));
(table_name, action_type, record_id,
                                                 RETURN NEW;
                                              ELSIF (TG OP = 'UPDATE') THEN
old values)
                                                           INTO
                                                                    log table
       VALUES ('details', 'DELETE',
                                                 INSERT
OLD.detail id, to jsonb(OLD));
                                          (table name, action type, record id,
       RETURN OLD;
                                          old_values, new_values)
                                                  VALUES ('invoice', 'UPDATE',
   END IF;
END;
                                          OLD.invoice id,
                                                               to jsonb(OLD),
$$;
                                          to jsonb(NEW));
                                                  RETURN NEW;
                                              ELSIF (TG OP = 'DELETE') THEN
                            FUNCTION
                                                           INTO log table
                                                 INSERT
public.log details changes()
                             OWNER
                                          (table name, action_type, record_id,
TO postgres;
                                          old values)
                            FUNCTION
                                                 VALUES ('invoice', 'DELETE',
public.log_employee_changes()
                                          OLD.invoice id, to jsonb(OLD));
RETURNS trigger
                                                 RETURN OLD;
   LANGUAGE plpgsql
                                              END IF;
   AS $$
                                          END;
BEGIN
                                          $$;
    IF (TG OP = 'INSERT') THEN
        INSERT
                  INTO
                                          ALTER
                          log table
(table name, action type, record id,
                                          public.log invoice changes() OWNER
new values)
                                          TO postgres;
       VALUES
                        ('employee',
'INSERT',
                   NEW.employee id,
                                          CREATE
                                                                      FUNCTION
to jsonb(NEW));
                                          public.log invoice detail changes()
       RETURN NEW;
                                          RETURNS trigger
    ELSIF (TG OP = 'UPDATE') THEN
                                             LANGUAGE plpqsql
       INSERT
                 INTO
                          log table
                                             AS $$
(table name, action type, record id,
                                          BEGIN
old values, new values)
                                             IF (TG OP = 'INSERT') THEN
       VALUES
                        ('employee',
                                                  INSERT
                                                           INTO log table
'UPDATE',
                    OLD.employee id,
                                          (table name, action type, record id,
to_jsonb(OLD), to_jsonb(NEW));
                                          new values)
       RETURN NEW;
    ELSIF (TG OP = 'DELETE') THEN
```

Изм.	Лист	№ докум.	Подпись	Дата

```
('invoice detail',
                                                                     FUNCTION
                                         ALTER
'INSERT',
                     NEW.invoiceID,
                                         public.log invoice employee changes(
                                         ) OWNER TO postgres;
to_jsonb(NEW));
       RETURN NEW;
   ELSIF (TG OP = 'UPDATE') THEN
                                         CREATE
                                                                     FUNCTION
                                         public.log_rack_changes()
       INSERT
                 INTO log table
                                                                     RETURNS
(table_name, action_type, record_id,
                                         trigger
old_values, new_values)
                                             LANGUAGE plpgsql
       VALUES
                 ('invoice detail',
                                             AS $$
'UPDATE',
                     OLD.invoiceID,
                                         BEGIN
to jsonb(OLD), to jsonb(NEW));
                                             IF (TG OP = 'INSERT') THEN
                                                 INSERT INTO log table
       RETURN NEW;
   ELSIF (TG OP = 'DELETE') THEN
                                         (table name, action type, record id,
        INSERT
                INTO log table
                                         new values)
                                                        ('rack', 'INSERT',
(table name, action type, record id,
                                                 VALUES
                                         NEW.rack id, to jsonb(NEW));
       VALUES
                  ('invoice detail',
                                                 RETURN NEW;
                                             ELSIF (TG OP = 'UPDATE') THEN
'DELETE',
                     OLD.invoiceID,
to_jsonb(OLD));
                                                 INSERT
                                                           INTO log table
       RETURN OLD;
                                          (table name, action type, record id,
   END IF;
                                         old values, new values)
END;
                                                 VALUES ('rack',
                                                                   'UPDATE',
                                         OLD.rack id,
                                                               to jsonb(OLD),
$$;
                                         to jsonb(NEW));
                                                 RETURN NEW;
                                             ELSIF (TG OP = 'DELETE') THEN
public.log invoice detail changes()
                                                           INTO log_table
OWNER TO postgres;
                                                INSERT
                                          (table_name, action_type, record_id,
                           FUNCTION
CREATE
                                         old values)
public.log invoice employee changes(
                                                 VALUES ('rack',
                                                                   'DELETE',
) RETURNS trigger
                                         OLD.rack id, to jsonb(OLD));
   LANGUAGE plpgsql
                                                 RETURN OLD;
   AS $$
                                             END IF;
BEGIN
                                         END;
    IF (TG OP = 'INSERT') THEN
                                         $$;
       INSERT
                 INTO log_table
                                                                     FUNCTION
(table_name, action_type, record_id,
                                         ALTER
                                         public.log rack changes() OWNER TO
new values)
       VALUES ('invoice employee',
                                         postgres;
'INSERT',
                    NEW.invoiceID,
to jsonb(NEW));
                                         CREATE
                                                                     FUNCTION
       RETURN NEW;
                                         public.log_room_changes()
                                                                     RETURNS
    ELSIF (TG OP = 'UPDATE') THEN
                                         trigger
                 INTO
       INSERT
                         log table
                                             LANGUAGE plpgsql
(table_name, action_type, record_id,
                                             AS $$
old values, new values)
                                         BEGIN
       VALUES ('invoice employee',
                                             IF (TG OP = 'INSERT') THEN
                                                 INSERT
'UPDATE',
                                                           INTO
                     OLD.invoiceID,
                                                                   log table
to jsonb(OLD), to jsonb(NEW));
                                          (table_name, action_type, record_id,
       RETURN NEW;
                                         new values)
                                                         ('room', 'INSERT',
   ELSIF (TG OP = 'DELETE') THEN
                                                 VALUES
       INSERT
                 INTO log table
                                         NEW.room id, to jsonb(NEW));
(table_name, action_type, record_id,
                                                 RETURN NEW;
                                             ELSIF (TG OP = 'UPDATE') THEN
old values)
                                                           INTO log_table
       VALUES ('invoice employee',
                                                 INSERT
'DELETE',
                    OLD.invoiceID,
                                          (table name, action type, record id,
to jsonb(OLD));
                                         old_values, new_values)
       RETURN OLD;
                                                 VALUES ('room',
                                                                   'UPDATE',
                                         OLD.room id,
   END IF;
                                                              to jsonb(OLD),
END:
                                         to jsonb(NEW));
$$;
                                                 RETURN NEW;
                                             ELSIF (TG OP = 'DELETE') THEN
```

 Изм.
 Лист
 № докум.
 Подпись
 Дата

```
INSERT
                INTO
                         log table
                                                RETURN NEW;
(table name, action type, record id,
                                             ELSIF (TG OP = 'UPDATE') THEN
                                                          INTO log_table
                                                INSERT
old values)
       VALUES
               ('room', 'DELETE',
                                         (table name, action type, record id,
OLD.room id, to jsonb(OLD));
                                         old_values, new_values)
                                                 VALUES
        RETURN OLD;
                                                                ('warehouse',
   END IF;
                                         'UPDATE',
                                                           OLD.warehouse id,
END:
                                         to jsonb(OLD), to jsonb(NEW));
$$;
                                                 RETURN NEW;
                                             ELSIF (TG OP = 'DELETE') THEN
ALTER
                           FUNCTION
                                                 INSERT
                                                           INTO log table
public.log_room_changes() OWNER TO
                                         (table name, action type, record id,
postgres;
                                         old values)
                                                 VALUES
                                                               ('warehouse',
                           FUNCTION
                                         'DELETE',
                                                           OLD.warehouse id,
CREATE
                                         to_jsonb(OLD));
public.log_shelf_changes()
                          RETURNS
                                                 RETURN OLD;
   LANGUAGE plpgsql
                                             END IF;
   AS $$
                                         END;
BEGIN
                                         $$;
    IF (TG OP = 'INSERT') THEN
       INSERT INTO log table
                                         ALTER
                                                                     FUNCTION
                                         public.log warehouse changes() OWNER
(table name, action type, record id,
new_values)
                                         TO postgres;
       VALUES ('shelf', 'INSERT',
NEW.shelf id, to jsonb(NEW));
                                         CREATE
                                                                     FUNCTION
       RETURN NEW;
                                         public.update invoice details view()
   ELSIF (TG OP = 'UPDATE') THEN
                                         RETURNS trigger
                 INTO
                                            LANGUAGE plpgsql
       INSERT
                         log table
                                             AS $$
(table name, action type, record id,
old values, new values)
                                         DECLARE
                                             v_updated BOOLEAN := FALSE;
       VALUES ('shelf', 'UPDATE',
OLD.shelf id,
                    to jsonb(OLD),
                                         BEGIN
to_jsonb(NEW));
                                            ΙF
                                                  NEW.type invoice text
                                         DISTINCT FROM OLD.type_invoice_text
       RETURN NEW;
   ELSIF (TG_OP = 'DELETE') THEN
                INTO log table
       INSERT
                                                 UPDATE invoice SET
                                                    type_invoice
(table_name, action_type, record_id,
                                         convert text to boolean (NEW.type inv
old values)
       VALUES ('shelf', 'DELETE',
                                         oice text, 'type')
OLD.shelf id, to jsonb(OLD));
                                                 WHERE
                                                           invoice id
       RETURN OLD;
                                         NEW.invoice id;
                                                 v_updated := TRUE;
   END IF;
END;
                                             END IF;
$$;
                                             IF NEW.status text IS DISTINCT
                                         FROM OLD.status text THEN
                                                UPDATE invoice SET
public.log shelf changes() OWNER TO
                                                     status
                                         convert text to boolean (NEW.status t
postgres;
                                         ext, 'status')
CREATE
                           FUNCTION
                                                 WHERE
                                                            invoice id
public.log_warehouse_changes()
                                         NEW.invoice id;
RETURNS trigger
                                                v updated := TRUE;
   LANGUAGE plpgsql
                                             END IF;
   AS $$
                                             IF NOT v_updated AND (
BEGIN
                                                NEW.invoice id IS DISTINCT
    IF (TG OP = 'INSERT') THEN
                                         FROM OLD.invoice id OR
       INSERT
                 INTO log_table
                                                NEW.counteragent name
                                                                           TS
(table name, action type, record id,
                                         DISTINCT FROM OLD.counteragent name
new values)
                      ('warehouse',
       VALUES
                                                 NEW.date time IS DISTINCT
'INSERT',
                  NEW.warehouse id,
                                         FROM OLD.date time OR
to jsonb(NEW));
```

```
NEW.type detail IS DISTINCT
                                           END IF;
                                            IF OLD.status IS NOT DISTINCT
FROM OLD.type detail OR
       NEW.quantity IS
                           DISTINCT
                                        FROM NEW.status THEN
                                                RETURN NEW;
FROM OLD.quantity OR
       NEW.responsible_last_name IS
                                            END IF;
                                            UPDATE invoice SET status
DISTINCT
                               FROM
                                        NEW.status
OLD.responsible last name OR
       NEW.responsible_first_name
                                         WHERE
                                                         invoice id
            DISTINCT
                               FROM
                                        NEW.invoice_id;
                                           RETURN NEW;
OLD.responsible_first_name OR
       NEW.responsible patronymic
                                        END;
            DISTINCT
                              FROM
                                        $$;
TS
OLD.responsible patronymic
    ) THEN
                                        ALTER
                                                                    FUNCTION
       RAISE EXCEPTION 'Разрешено
                                        public.update invoice status() OWNER
                                        TO postgres;
           только
type_invoice_text и status_text';
   END IF;
                                        CREATE
                                                                    FUNCTION
                                        public.update warehouse_details_view
   RETURN NEW;
                                         () RETURNS trigger
END;
                                            LANGUAGE plpgsql
$$;
                                            AS $$
                                         BEGIN
public.update invoice details view()
                                            UPDATE details
                                                       type detail
OWNER TO postgres;
                                        NEW.type detail, weight = NEW.weight
                                            WHERE detail id = OLD.detail_id;
                           FUNCTION
public.update invoice status()
                                            UPDATE shelf
RETURNS trigger
                                            SET
                                                       shelf number
   LANGUAGE
                         SECURITY
                                        NEW.shelf number
               plpgsql
DEFINER
                                            WHERE shelf id = (SELECT shelfID
                                         FROM details WHERE detail id =
   AS $$
BEGIN
                                         OLD.detail id);
    IF TG_OP = 'UPDATE' AND (
                                            UPDATE rack
       OLD.invoice id IS DISTINCT
                                            SET
                                                       rack number
FROM NEW.invoice id OR
                                        NEW.rack number
                                           WHERE rack id = (SELECT rackID
       OLD.counteragent name
                                         FROM shelf WHERE shelf id = (SELECT
DISTINCT FROM NEW.counteragent name
                                         shelfID FROM details WHERE detail id
                                         = OLD.detail id));
       OLD.date time
                     IS DISTINCT
FROM NEW.date time OR
                                            UPDATE room
       OLD.type invoice IS DISTINCT
                                            SET
                                                        room number
FROM NEW.type_invoice OR
                                         NEW.room number
       OLD.type detail IS DISTINCT
                                            WHERE room id = (SELECT roomID
                                         FROM rack WHERE rack id = (SELECT
FROM NEW.type_detail OR
       OLD.quantity
                     IS
                           DISTINCT
                                         rackID FROM shelf WHERE shelf id =
FROM NEW.quantity OR
                                         (SELECT shelfID FROM details WHERE
                                         detail id = OLD.detail id)));
       OLD.responsible last name IS
                               FROM
                                            UPDATE warehouse
                                            SET
NEW.responsible last name OR
                                                    warehouse number
       OLD.responsible first name
                                        NEW.warehouse number
            DISTINCT
                                           WHERE warehouse id = (SELECT
NEW.responsible_first_name OR
                                        warehouseID FROM room WHERE room id
       OLD.responsible patronymic
                                         = (SELECT roomID FROM rack WHERE
                                         rack id = (SELECT rackID FROM shelf
            DISTINCT
                              FROM
NEW.responsible patronymic OR
                                        WHERE shelf_id = (SELECT shelfID
       OLD.responsible id
                                 IS
                                        FROM details WHERE detail id =
                                        OLD.detail_id))));
DISTINCT FROM NEW.responsible id
    ) THEN
                                            RETURN NEW;
       RAISE EXCEPTION 'Разрешено
                                        END:
                                         $$;
обновлять только поле status.
         изменить другие поля
Попытка
запрещена.';
```

Изм.	Лист	№ докум.	Подпись	Дат

```
ALTER
                           FUNCTION
                                        ALTER
                                                                    SEOUENCE
public.update warehouse details view
                                       public.details detail id seq
() OWNER TO postgres;
                                        BY public.details.detail id;
SET default_tablespace = '';
                                        CREATE
                                                                    SEOUENCE
                                      public.details_shelfid_seq
SET default table access method =
                                           AS integer
heap;
                                            START WITH 1
CREATE TABLE public.counteragent (
                                            INCREMENT BY 1
   counteragent id integer NOT
                                           NO MINVALUE
                                            NO MAXVALUE
                                            CACHE 1;
   counteragent name
                        character
varying (128) NOT NULL,
                          character
                                      ALTER
                                                                    SEQUENCE
  contact person
varying(128) NOT NULL,
                                        public.details shelfid seq OWNER TO
   phone number bigint NOT NULL,
   address text NOT NULL
                                        ALTER
                                                                    SEQUENCE
                                        public.details shelfid seq OWNED BY
                                        public.details.shelfid;
ALTER TABLE public.counteragent
OWNER TO postgres;
                                        CREATE TABLE public.employee (
                                            employee id integer NOT NULL,
                           SEQUENCE
                                            employee role
public.counteragent_counteragent_id_
                                        varying(25) NOT NULL,
                                            last name character varying (35)
   AS integer
                                        NOT NULL,
   START WITH 1
                                            first name character varying (35)
   INCREMENT BY 1
                                        NOT NULL,
   NO MINVALUE
                                            patronymic character varying(35)
   NO MAXVALUE
                                        NOT NULL
   CACHE 1;
                                        );
                           SEOUENCE
                                        ALTER TABLE public.employee OWNER TO
public.counteragent_counteragent_id_
                                        postgres;
seq OWNER TO postgres;
                           SEOUENCE
                                        CREATE
                                                                    SEQUENCE
ALTER
public.counteragent counteragent id
                                        public.employee_employee_id_seq
                                ΒŸ
               OWNED
                                          AS integer
                                            START WITH 1
public.counteragent.counteragent_id;
                                            INCREMENT BY 1
CREATE TABLE public.details (
                                            NO MINVALUE
   detail id integer NOT NULL,
                                            NO MAXVALUE
    shelfid integer NOT NULL,
                                            CACHE 1;
   weight double precision
                               NOT
NULL,
                                        ALTER
   type detail text NOT NULL
                                        public.employee employee id seq
                                        OWNER TO postgres;
                                                                    SEQUENCE
                                        ALTER
ALTER TABLE public.details OWNER TO
                                        public.employee employee id seq
                                                                          ВҮ
postgres;
                                        public.employee.employee id;
CREATE
                           SEQUENCE
public.details detail id seq
                                        CREATE TABLE public.invoice (
   AS integer
                                            invoice id integer NOT NULL,
   START WITH 1
                                             counteragentid integer NOT NULL,
                                            date_time timestamp without time
   INCREMENT BY 1
   NO MINVALUE
                                        zone NOT NULL,
                                            type invoice boolean NOT NULL,
   NO MAXVALUE
   CACHE 1;
                                             status boolean NOT NULL
                                         );
                           SEOUENCE
ALTER
public.details detail id seq OWNER
                                        ALTER TABLE public.invoice OWNER TO
TO postgres;
                                        postgres;
```

```
invoiceid integer NOT NULL,
CREATE
                            SEOUENCE
                                              responsible integer NOT NULL,
public.invoice_counteragentid_seq
                                              granted access integer NOT NULL,
                                              when granted timestamp without
   AS integer
    START WITH 1
                                          time zone NOT NULL
    INCREMENT BY 1
                                          );
   NO MINVALUE
   NO MAXVALUE
                                          ALTER TABLE public.invoice employee
                                          OWNER TO postgres;
   CACHE 1:
ALTER
                            SEQUENCE
                                         CREATE
                                                                          VIEW
public.invoice_counteragentid_seq
                                          public.invoice details view AS
OWNER TO postgres;
                                           SELECT inv.invoice id,
                            SEQUENCE
ALTER
                                              ca.counteragent name,
public.invoice counteragentid seq
                                              inv.date time,
                                  BY
                                                  CASE
                                                              inv.type_invoice
public.invoice.counteragentid;
                                                      WHEN
                                          THEN 'Выгрузка'::text
CREATE TABLE public.invoice detail (
                                                      ELSE 'Отгрузка'::text
    invoiceid integer NOT NULL,
                                                  END AS type invoice text,
    detailid integer NOT NULL,
    quantity integer NOT NULL
                                                      WHEN
                                                             inv.status
                                                                         THEN
                                          'Завершено'::text
                                                      ELSE 'В процессе'::text
ALTER TABLE public.invoice detail
                                                  END AS status text,
OWNER TO postgres;
                                              det.type detail,
                                              invd.quantity,
CREATE
                            SEOUENCE
                                              emp.last name
                                                                            AS
public.invoice detail detailid seq
                                          responsible last name,
   AS integer
                                              emp.first name
                                                                            AS
    START WITH 1
                                          responsible first name,
   INCREMENT BY 1
                                              emp.patronymic
                                                                            AS
   NO MINVALUE
                                          responsible patronymic,
   NO MAXVALUE
                                              emp.employee id
                                                                            AS
   CACHE 1;
                                          responsible id,
                                              inv.status AS status bool,
                            SEQUENCE
                                              inv.type_invoice
                                                                            AS
                                          type invoice bool
public.invoice detail detailid seq
OWNER TO postgres;
                                             FROM ((((public.invoice inv
                            SEQUENCE
                                               JOIN public.invoice detail invd
ALTER
                                                    ((inv.invoice id
public.invoice detail detailid seq
                                          ON
                                          invd.invoiceid)))
public.invoice detail.detailid;
                                               JOIN public.details det
                                          ((invd.detailid = det.detail id)))
                            SEQUENCE
                                              JOIN public.invoice_employee
public.invoice detail invoiceid seq
                                          inv emp ON ((inv.invoice id
                                          inv emp.invoiceid)))
   AS integer
    START WITH 1
                                               JOIN public.employee emp
    INCREMENT BY 1
                                          ((inv emp.responsible
    NO MINVALUE
                                          emp.employee id)))
    NO MAXVALUE
                                               JOIN public.counteragent ca ON
    CACHE 1;
                                          ((inv.counteragentid
                                          ca.counteragent id)));
ALTER
                            SEQUENCE
public.invoice detail invoiceid seq
                                          ALTER
                                                                          VIEW
OWNER TO postgres;
                                          public.invoice details view OWNER TO
ALTER
                            SEOUENCE
                                          postgres;
public.invoice detail invoiceid seq
OWNED
                                          CREATE
                                                                      SEOUENCE
public.invoice detail.invoiceid;
                                          public.invoice employee granted acce
                                          ss seq
CREATE TABLE public.invoice employee
                                              AS integer
                                              START WITH 1
```

```
INCREMENT BY 1
                                          ALTER
                                                                      SEQUENCE
   NO MINVALUE
                                          public.invoice invoice id seq OWNED
   NO MAXVALUE
                                          BY public.invoice.invoice id;
   CACHE 1;
                                          CREATE TABLE public.log table (
                                              log_id integer NOT NULL,
ALTER
                            SEOUENCE
public.invoice employee granted acce
                                              table name character varying (50)
ss seq OWNER TO postgres;
                                          NOT NULL,
                            SEQUENCE
                                             action_type
                                                                     character
public.invoice_employee_granted_acce
                                          varying (20) NOT NULL,
                 OWNED
                                             record id integer,
public.invoice employee.granted acce
                                             action time timestamp without
                                          time zone DEFAULT CURRENT TIMESTAMP,
                                             old values jsonb,
CREATE
                            SEOUENCE
                                              new values jsonb
public.invoice employee invoiceid se
    AS integer
                                          ALTER TABLE public.log table OWNER
    START WITH 1
                                          TO postgres;
    INCREMENT BY 1
   NO MINVALUE
                                          CREATE
   NO MAXVALUE
                                          public.log_table_log_id_seq
   CACHE 1;
                                             AS integer
                                              START WITH 1
                            SEQUENCE
                                             INCREMENT BY 1
public.invoice employee invoiceid se
                                             NO MINVALUE
q OWNER TO postgres;
                                             NO MAXVALUE
                            SEQUENCE
                                              CACHE 1;
ALTER
public.invoice_employee_invoiceid_se
               OWNED
                                          ALTER
                                                                      SEOUENCE
                                          public.log table log id seq OWNER TO
public.invoice employee.invoiceid;
                                          postgres;
                            SEOUENCE
                                         ALTER
                                                                      SEOUENCE
                                          public.log_table_log_id_seq OWNED BY
public.invoice_employee_responsible_
                                          public.log_table.log_id;
seq
    AS integer
    START WITH 1
                                          CREATE TABLE public.rack (
    INCREMENT BY 1
                                              rack id integer NOT NULL,
                                              roomid integer NOT NULL,
   NO MINVALUE
   NO MAXVALUE
                                              rack number integer NOT NULL
   CACHE 1;
                                          );
                            SEQUENCE
                                          ALTER TABLE public.rack OWNER TO
public.invoice_employee_responsible_
                                          postgres;
seq OWNER TO postgres;
                            SEQUENCE
                                          CREATE
                                                                      SEQUENCE
public.invoice employee responsible
                                          public.rack rack id seq
               OWNED
                                            AS integer
                                              START WITH 1
public.invoice employee.responsible;
                                              INCREMENT BY 1
CREATE
                            SEQUENCE
                                              NO MINVALUE
public.invoice invoice id seq
                                              NO MAXVALUE
   AS integer
                                              CACHE 1;
    START WITH 1
    INCREMENT BY 1
                                          ALTER
                                                                      SEQUENCE
   NO MINVALUE
                                          public.rack rack id seq OWNER
                                                                            TO
   NO MAXVALUE
                                          postgres;
   CACHE 1;
                                          ALTER
                                                                      SEOUENCE
                                          public.rack rack id seq OWNED
ALTER
                            SEOUENCE
                                          public.rack.rack id;
public.invoice invoice id seq OWNER
TO postgres;
                                          CREATE
                                                                      SEOUENCE
                                          public.rack_roomid_seq
```

KP.350000.000

```
AS integer
    START WITH 1
                                         CREATE
                                                                     SEOUENCE
    INCREMENT BY 1
                                         public.shelf rackid seq
   NO MINVALUE
                                             AS integer
   NO MAXVALUE
                                             START WITH 1
                                             INCREMENT BY 1
   CACHE 1;
                                             NO MINVALUE
ALTER
                           SEQUENCE
                                            NO MAXVALUE
                         OWNER TO
public.rack roomid seq
                                            CACHE 1;
postgres;
                                        ALTER
                           SEQUENCE
                                                                     SEQUENCE
ALTER
public.rack_roomid seq
                         OWNED BY
                                        public.shelf rackid seq OWNER TO
public.rack.roomid;
                                         postgres;
                                         ALTER
                                                                     SEQUENCE
CREATE TABLE public.room (
                                        public.shelf rackid seq
                                                                   OWNED BY
    room id integer NOT NULL,
                                        public.shelf.rackid;
    warehouseid integer NOT NULL,
    room number integer NOT NULL
                                        CREATE
                                                                     SEQUENCE
                                         public.shelf shelf id seq
                                            AS integer
ALTER TABLE public.room OWNER TO
                                             START WITH 1
                                             INCREMENT BY 1
postgres;
                                             NO MINVALUE
CREATE
                           SEQUENCE
                                            NO MAXVALUE
public.room room id seq
                                             CACHE 1;
   AS integer
   START WITH 1
                                                                     SEQUENCE
   INCREMENT BY 1
                                         public.shelf_shelf_id_seq OWNER TO
   NO MINVALUE
                                         postgres;
   NO MAXVALUE
                                         ALTER
                                                                     SEOUENCE
                                         public.shelf shelf id seq OWNED BY
   CACHE 1;
                                         public.shelf.shelf id;
                           SEOUENCE
                         OWNER TO
                                         CREATE TABLE public.warehouse (
public.room_room_id_seq
                                             warehouse id integer NOT NULL,
postgres;
                           SEQUENCE
                                             warehouse number integer NOT
ALTER
                                         NULL,
public.room_room_id_seq
                         OWNED BY
public.room.room id;
                                             address text NOT NULL
                                         );
CREATE
                           SEQUENCE
                                         ALTER TABLE public.warehouse OWNER
public.room warehouseid seq
   AS integer
                                         TO postgres;
    START WITH 1
    INCREMENT BY 1
                                         CREATE
                                                                         VIEW
    NO MINVALUE
                                         public.warehouse details view AS
   NO MAXVALUE
                                          SELECT w.warehouse number,
   CACHE 1;
                                             r.room number,
                                             rk.rack number,
                           SEQUENCE
                                             s.shelf number,
public.room warehouseid seq OWNER TO
                                             d.type detail,
postgres;
                                             d.weight,
ALTER
                           SEQUENCE
                                             d.detail id
public.room warehouseid seq OWNED BY
                                            FROM ((((public.warehouse w
                                                     public.room r
public.room.warehouseid;
                                              JOIN
                                         ((w.warehouse id = r.warehouseid)))
CREATE TABLE public.shelf (
                                              JOIN public.rack rk ON
    shelf id integer NOT NULL,
                                         ((r.room id = rk.roomid)))
    rackid integer NOT NULL,
                                             JOIN public.shelf
                                                                          ON
                                         ((rk.rack_id = s.rackid)))
    shelf number integer NOT NULL
                                              JOIN public.details d
                                                                          OM
);
                                         ((s.shelf id = d.shelfid)));
ALTER TABLE public.shelf OWNER TO
postgres;
```

```
ALTER
                              VIEW
                                        ALTER
                                                       TABLE
                                                                       ONLY
public.warehouse details view OWNER
                                        public.invoice_employee ALTER COLUMN
                                        responsible SET DEFAULT
TO postgres;
                                        nextval('public.invoice employee res
CREATE
                          SEOUENCE
                                       ponsible_seq'::regclass);
public.warehouse warehouse id seq
                                        ALTER
                                                       TABLE
                                                                      ONLY
                                        public.invoice_employee ALTER COLUMN
   AS integer
                                        granted_access SET DEFAULT
   START WITH 1
   INCREMENT BY 1
                                        nextval('public.invoice employee gra
   NO MINVALUE
                                        nted access seq'::regclass);
                                        ALTER TABLE ONLY public.log table
   NO MAXVALUE
                                        ALTER COLUMN log_id SET DEFAULT
   CACHE 1;
                                        nextval('public.log_table_log_id_seq
ALTER
                           SEQUENCE
                                        '::regclass);
                                        ALTER TABLE ONLY public.rack ALTER
public.warehouse warehouse id seq
OWNER TO postgres;
                                        COLUMN rack id SET DEFAULT
                           SEQUENCE
                                       nextval('public.rack_rack_id_seq'::r
                                        egclass);
public.warehouse_warehouse_id_seq
                                BY
                                        ALTER TABLE ONLY public.rack ALTER
public.warehouse.warehouse id;
                                        COLUMN roomid SET DEFAULT
                                        nextval('public.rack_roomid_seq'::re
ALTER TABLE ONLY public.counteragent
                                        gclass);
                                        ALTER TABLE ONLY public.room ALTER
ALTER COLUMN counteragent id SET
DEFAULT
                                        COLUMN room id SET
nextval('public.counteragent_counter
                                        nextval('public.room_room_id_seq'::r
agent id seq'::regclass);
                                        egclass);
ALTER TABLE ONLY public.details
                                        ALTER TABLE ONLY public.room ALTER
ALTER COLUMN detail id SET DEFAULT
                                        COLUMN warehouseid SET DEFAULT
nextval('public.details detail id se
                                        nextval('public.room warehouseid seq
                                        '::regclass);
q'::regclass);
                                        ALTER TABLE ONLY public.shelf ALTER
ALTER TABLE ONLY public.details
ALTER COLUMN shelfid SET DEFAULT
                                                shelf id SET DEFAULT
                                        COLUMN
                                        nextval('public.shelf shelf id seq':
nextval('public.details shelfid seq'
::regclass);
                                        :regclass);
ALTER TABLE ONLY public.employee
                                        ALTER TABLE ONLY public.shelf ALTER
ALTER COLUMN employee id SET DEFAULT
                                        COLUMN rackid SET DEFAULT
nextval('public.employee_employee_id
                                        nextval('public.shelf_rackid_seq'::r
seq'::regclass);
                                        egclass);
ALTER TABLE ONLY public.invoice
                                        ALTER TABLE ONLY public.warehouse
ALTER COLUMN invoice id SET DEFAULT
                                              COLUMN warehouse_id SET
                                        ALTER
nextval('public.invoice invoice id s
                                        DEFAULT
eq'::regclass);
                                        nextval('public.warehouse warehouse
ALTER TABLE ONLY public.invoice
                                        id seq'::regclass);
ALTER
      COLUMN counteragentid SET
nextval('public.invoice counteragent
                                        pg catalog.setval('public.counterage
id seq'::regclass);
                                        nt counteragent id seq', 6, true);
              TABLE
public.invoice_detail ALTER COLUMN
                                        pg catalog.setval('public.details de
               SET
invoiceid
                                        tail_id_seq', 47, true);
nextval('public.invoice_detail_invoi
                                        SELECT
                                        pg_catalog.setval('public.details sh
ceid seq'::regclass);
ALTER
              TABLE
                              ONLY
                                        elfid seq', 1, false);
public.invoice_detail ALTER COLUMN
                                        SELECT
detailid
               SET
                      DEFAULT
                                        pg catalog.setval('public.employee e
nextval('public.invoice detail detai
                                        mployee id seq', 6, true);
lid seq'::regclass);
                                        SELECT
ALTER
              TABLE
                              ONLY
                                        pg catalog.setval('public.invoice co
public.invoice_employee ALTER COLUMN
                                        unteragentid seq', 1, false);
                                        SELECT
invoiceid
                SET
                       DEFAULT
                                        pg catalog.setval('public.invoice de
nextval('public.invoice employee inv
oiceid seq'::regclass);
                                        tail detailid seq', 1, false);
```

Изм.	Лист	№ докум.	Подпись	Дата

```
SELECT
                                            ADD
                                                                 CONSTRAINT
pg catalog.setval('public.invoice de
                                       invoice employee unique
                                                                    UNIOUE
tail invoiceid seq', 1, false);
                                        (invoiceid, responsible);
SELECT
pg catalog.setval('public.invoice em
                                        ALTER TABLE ONLY public.invoice
                                           ADD CONSTRAINT invoice_pkey
ployee granted access seq',
                                        PRIMARY KEY (invoice id);
SELECT
pg catalog.setval('public.invoice em
                                        ALTER TABLE ONLY public.log table
ployee_invoiceid_seq', 1, false);
                                           ADD CONSTRAINT log table pkey
                                        PRIMARY KEY (log id);
pg catalog.setval('public.invoice em
ployee_responsible_seq', 1, false);
                                        ALTER TABLE ONLY public.rack
                                            ADD CONSTRAINT rack pkey PRIMARY
SELECT
pg catalog.setval('public.invoice in
                                        KEY (rack id);
voice id seq', 13, true);
                                        ALTER TABLE ONLY public.room
pg catalog.setval('public.log_table_
                                           ADD CONSTRAINT room pkey PRIMARY
log_id_seq', 1954, true);
                                        KEY (room id);
pg catalog.setval('public.rack_rack_
                                        ALTER TABLE ONLY public.shelf
id seq', 126, true);
                                           ADD CONSTRAINT shelf pkey
                                        PRIMARY KEY (shelf id);
pg catalog.setval('public.rack roomi
d seq', 1, false);
                                        ALTER TABLE ONLY public.warehouse
SELECT
                                           ADD CONSTRAINT unique address
pg catalog.setval('public.room room
                                        UNIQUE (address);
id seq', 26, true);
                                        ALTER TABLE ONLY public.warehouse
                                           ADD CONSTRAINT warehouse pkey
pg catalog.setval('public.room wareh
ouseid seq', 1, false);
                                        PRIMARY KEY (warehouse id);
pg_catalog.setval('public.shelf rack
                                        CREATE TRIGGER counteragent changes
                                        AFTER INSERT OR DELETE OR UPDATE ON
id seq', 1, false);
                                        public.counteragent FOR EACH ROW
pg catalog.setval('public.shelf shel
                                        EXECUTE
                                                                  FUNCTION
f id seq', 625, true);
                                        public.log_counteragent_changes();
                                        CREATE TRIGGER delete_related_data
SELECT
                                        BEFORE DELETE ON public.rack FOR
pg catalog.setval('public.warehouse
                                               ROW EXECUTE
warehouse id seq', 9, true);
                                                                 FUNCTION
                                        EACH
                                        public.delete related data();
                                        CREATE TRIGGER delete related data
ALTER TABLE ONLY public.counteragent
                                        BEFORE DELETE ON public.room FOR
   ADD CONSTRAINT counteragent pkey
PRIMARY KEY (counteragent id);
                                        EACH
                                               ROW EXECUTE
                                                                FUNCTION
                                        public.delete_related_data();
ALTER TABLE ONLY public.details
                                        CREATE TRIGGER delete related data
                                        BEFORE DELETE ON public.shelf FOR
   ADD CONSTRAINT details pkey
PRIMARY KEY (detail id);
                                        EACH
                                               ROW
                                                      EXECUTE
                                                                  FUNCTION
                                        public.delete related data();
ALTER TABLE ONLY public.employee
                                        CREATE TRIGGER delete related data
                                        BEFORE DELETE ON public.warehouse
   ADD CONSTRAINT employee pkey
                                        FOR EACH ROW EXECUTE FUNCTION
PRIMARY KEY (employee id);
                                        public.delete related data();
ALTER
               TABLE
                              ONLY
                                        CREATE TRIGGER details changes AFTER
public.invoice detail
                                        INSERT OR DELETE OR UPDATE ON
                                        public.details FOR EACH ROW EXECUTE
   ADD
                        CONSTRAINT
invoice detail unique
                         UNIQUE
                                        FUNCTION
                                        public.log_details_changes();
(invoiceid, detailid);
                                        CREATE TRIGGER employee_changes
                              ONLY
                                        AFTER INSERT OR DELETE OR UPDATE ON
ALTER
               TABLE
                                        public.employee FOR EACH ROW EXECUTE
public.invoice employee
                                        FUNCTION
                                        public.log_employee_changes();
                                                                            Лист
```

Изм.

Лист

№ докум.

Подпись Дата

KP.350000.000

114

```
INSERT OR DELETE OR UPDATE ON
                                      AFTER INSERT OR DELETE OR UPDATE ON
public.invoice FOR EACH ROW EXECUTE
                                      public.warehouse FOR EACH ROW
                                      EXECUTE
                                                               FUNCTION
FUNCTION
                                      public.log_warehouse_changes();
public.log_invoice_changes();
                          TRIGGER
invoice_detail_changes AFTER INSERT
                                      ALTER TABLE ONLY public.details
     DELETE OR UPDATE ON
                                                              CONSTRAINT
public.invoice_detail FOR EACH ROW
                                      details_shelfid_fkey FOREIGN KEY
                         FUNCTION
                                      (shelfid)
                                                              REFERENCES
public.log_invoice_detail_changes();
                                      public.shelf(shelf id);
CREATE
invoice details view delete INSTEAD
                                      ALTER TABLE ONLY public.invoice
       DELETE
                                                             CONSTRAINT
public.invoice_details_view FOR EACH
                                      invoice counteragentid fkey FOREIGN
ROW EXECUTE FUNCTION
                                      KEY (counteragentid) REFERENCES
public.delete_invoice_details_view()
                                      public.counteragent(counteragent id)
invoice_details view insert INSTEAD
                                      ALTER
                                                    TABLE
                                                                   ONLY
            INSERT
                                      public.invoice detail
public.invoice_details_view FOR EACH
                                                             CONSTRAINT
ROW EXECUTE FUNCTION
                                      invoice detail detailid fkey FOREIGN
                                      KEY (detailid) REFERENCES
public.insert invoice details view()
                                      public.details(detail id);
CREATE
                          TRIGGER
invoice details view update INSTEAD
                                                    TABLE
                                                                   ONLY
            UPDATE
                                      public.invoice detail
public.invoice details view FOR EACH
                                         ADD
                                      invoice detail invoiceid fkey
ROW EXECUTE
                     FUNCTION
                                      FOREIGN KEY (invoiceid) REFERENCES
public.update invoice details view()
                                      public.invoice(invoice id);
                          TRIGGER
invoice_employee_changes
                           AFTER
                                      ALTER
                                                    TABLE
                                                                   ONLY
INSERT OR DELETE OR UPDATE ON
                                      public.invoice_employee
public.invoice_employee FOR EACH ROW
                                         ADD
                                                              CONSTRAINT
                                      invoice_employee_granted_access_fkey
                        FUNCTION
                                      FOREIGN KEY (granted_access)
public.log invoice employee changes(
                                      REFERENCES
CREATE TRIGGER rack_changes AFTER
                                      public.employee(employee id);
INSERT OR DELETE OR UPDATE ON
public.rack FOR EACH ROW EXECUTE
                                      ALTER
                                                    TABLE
                                                                   ONLY
FUNCTION public.log_rack_changes();
                                      public.invoice_employee
CREATE TRIGGER room_changes AFTER
                                         ADD
                                                              CONSTRAINT
INSERT OR DELETE OR UPDATE ON
                                      invoice_employee_invoiceid_fkey
public.room FOR EACH ROW EXECUTE
                                      FOREIGN KEY (invoiceid) REFERENCES
FUNCTION public.log room changes();
                                      public.invoice(invoice id);
CREATE TRIGGER shelf changes AFTER
      OR DELETE OR UPDATE ON
                                                                   ONLY
public.shelf FOR EACH ROW EXECUTE
                                      public.invoice employee
FUNCTION public.log_shelf_changes();
                                         ADD
                                                              CONSTRAINT
                         TRIGGER
                                      invoice_employee_responsible_fkey
FOREIGN KEY (responsible) REFERENCES
DELETE ON public.invoice FOR EACH
                                      public.employee(employee id);
ROW EXECUTE
                         FUNCTION
public.delete invoice details view()
                                      ALTER TABLE ONLY public.rack
                                         ADD CONSTRAINT rack roomid fkey
                          TRIGGER
                                      FOREIGN KEY (roomid) REFERENCES
trg_delete_related_data
                          BEFORE
                                      public.room(room id);
DELETE ON public.warehouse FOR EACH
          EXECUTE
                                      ALTER TABLE ONLY public.room
                         FUNCTION
public.delete related data();
                                                              CONSTRAINT
                                      room_warehouseid_fkey FOREIGN KEY
```

CREATE TRIGGER invoice changes AFTER

Лист

№ докум.

Подпись Дата

KP.350000.000

CREATE TRIGGER warehouse changes

(warehouseid) REFERENCES	GRANT ALL ON FUNCTION
<pre>public.warehouse(warehouse_id);</pre>	<pre>public.log_rack_changes()</pre>
ALMED MADIE ONLY sublines half	warehouse_owner;
ALTER TABLE ONLY public.shelf ADD CONSTRAINT shelf rackid fkey	GRANT ALL ON FUNCTION public.log room changes() TO
FOREIGN KEY (rackid) REFERENCES	<pre>public.log_room_changes() TO warehouse owner;</pre>
public.rack(rack id);	GRANT ALL ON FUNCTION
pablic.idex(idex_id)/	public.log shelf changes() TO
GRANT USAGE ON SCHEMA public TO	warehouse owner;
warehouse owner;	GRANT ALL ON FUNCTION
GRANT ALL ON FUNCTION	<pre>public.log warehouse changes()</pre>
<pre>public.convert_text_to_boolean(text_</pre>	warehouse_owner;
<pre>value text, field_type text) TO</pre>	GRANT ALL ON FUNCTION
<pre>warehouse_owner;</pre>	<pre>public.update_invoice_details_view()</pre>
GRANT ALL ON FUNCTION	TO warehouse_owner;
<pre>public.delete_invoice_details_view()</pre>	GRANT ALL ON FUNCTION
TO warehouse_owner; GRANT ALL ON FUNCTION	<pre>public.update_invoice_status() TO warehouse owner;</pre>
public.delete related data() TO	GRANT ALL ON FUNCTION
warehouse owner;	public.update warehouse details view
GRANT ALL ON FUNCTION	() TO warehouse owner;
public.delete warehouse details() TO	GRANT SELECT, INSERT, DELETE, UPDATE ON
warehouse owner;	TABLE public.counteragent TO
GRANT ALL ON FUNCTION	<pre>warehouse_owner;</pre>
<pre>public.get_employee_id(p_last_name</pre>	GRANT SELECT ON TABLE
character varying, p_first_name	public.counteragent TO
character varying, p_patronymic	warehouse_manager;
character varying) TO	GRANT SELECT, USAGE ON SEQUENCE
warehouse_owner;	public.counteragent_counteragent_id_
GRANT ALL ON FUNCTION public.get employee id(p last name	<pre>seq TO warehouse_owner; GRANT SELECT,INSERT,DELETE,UPDATE ON</pre>
character varying, p first name	TABLE public.details TO
character varying, p_irist_name character varying, p patronymic	warehouse owner;
character varying) TO	GRANT SELECT, INSERT, DELETE, UPDATE ON
warehouse manager;	TABLE public.details TO
GRANT ALL ON FUNCTION	warehouse clerk;
<pre>public.insert_into_warehouse_details</pre>	GRANT SELECT ON TABLE public.details
() TO warehouse_owner;	TO warehouse_manager;
GRANT ALL ON FUNCTION	GRANT SELECT, USAGE ON SEQUENCE
<pre>public.insert_invoice_details_view()</pre>	<pre>public.details_detail_id_seq TO</pre>
TO warehouse_owner; GRANT ALL ON FUNCTION	<pre>warehouse_clerk; GRANT SELECT,USAGE ON SEQUENCE</pre>
public.insert invoice details view()	GRANT SELECT, USAGE ON SEQUENCE public.details detail id seq TO
TO warehouse manager;	warehouse owner;
GRANT ALL ON FUNCTION	GRANT SELECT, USAGE ON SEQUENCE
<pre>public.log counteragent changes() TO</pre>	public.details shelfid seq TO
warehouse_owner;	warehouse_owner;
GRANT ALL ON FUNCTION	GRANT SELECT, INSERT, DELETE, UPDATE ON
<pre>public.log_details_changes() TO</pre>	TABLE public.employee TO
warehouse_owner;	warehouse_owner;
GRANT ALL ON FUNCTION	GRANT SELECT, USAGE ON SEQUENCE
<pre>public.log_employee_changes() TO</pre>	public.employee_employee_id_seq TO
<pre>warehouse_owner; GRANT ALL ON FUNCTION</pre>	<pre>warehouse_owner; GRANT SELECT,USAGE ON SEQUENCE</pre>
public.log invoice changes() TO	public.employee employee id seq TO
warehouse owner;	warehouse manager;
GRANT ALL ON FUNCTION	GRANT SELECT ON TABLE public.invoice
<pre>public.log invoice detail changes()</pre>	TO warehouse owner;
TO warehouse_owner;	GRANT SELECT, UPDATE ON TABLE
GRANT ALL ON FUNCTION	<pre>public.invoice TO warehouse_clerk;</pre>
<pre>public.log_invoice_employee_changes(</pre>	GRANT SELECT, INSERT, DELETE, UPDATE ON
) TO warehouse_owner;	TABLE public.invoice TO
	<pre>warehouse_manager;</pre>

Изм.

Лист

№ докум.

Подпись

Дата

GRANT SELECT, USAGE ON SEQUENCE public.invoice counteragentid seq TO	GRANT SELECT, USAGE ON SEQUENCE public.log table log id seq TO
warehouse_owner; GRANT SELECT ON TABLE	<pre>warehouse_clerk; GRANT SELECT,USAGE ON SEQUENCE</pre>
<pre>public.invoice_detail TO</pre>	<pre>public.log_table_log_id_seq T0</pre>
warehouse_owner;	warehouse_manager;
GRANT SELECT, INSERT, DELETE, UPDATE ON TABLE public.invoice_detail TO	GRANT SELECT, USAGE ON SEQUENCE public.log_table_log_id_seq TO
warehouse_manager;	warehouse_owner;
GRANT SELECT ON TABLE	GRANT SELECT, INSERT, DELETE, UPDATE ON
public.invoice_detail TO	TABLE public.rack TO
warehouse_clerk; GRANT SELECT,USAGE ON SEQUENCE	<pre>warehouse_owner; GRANT SELECT,INSERT,DELETE,UPDATE ON</pre>
public.invoice detail detailid seq	TABLE public.rack TO
TO warehouse_owner;	warehouse_clerk;
GRANT SELECT, USAGE ON SEQUENCE	GRANT SELECT, USAGE ON SEQUENCE
<pre>public.invoice_detail_invoiceid_seq</pre>	public.rack_rack_id_seq TO
TO warehouse_manager;	warehouse_owner;
GRANT SELECT, USAGE ON SEQUENCE public.invoice detail invoiceid seq	GRANT SELECT, USAGE ON SEQUENCE public.rack rack id seq TO
TO warehouse owner;	warehouse clerk;
GRANT SELECT ON TABLE	GRANT SELECT, USAGE ON SEQUENCE
<pre>public.invoice_employee TO</pre>	<pre>public.rack_roomid_seq TO</pre>
<pre>warehouse_owner;</pre>	<pre>warehouse_owner;</pre>
GRANT SELECT, INSERT, DELETE, UPDATE ON	GRANT SELECT, INSERT, DELETE, UPDATE ON
TABLE public.invoice_employee TO	TABLE public.room TO
<pre>warehouse_manager; GRANT SELECT ON TABLE</pre>	<pre>warehouse_owner; GRANT SELECT,INSERT,DELETE,UPDATE ON</pre>
public.invoice employee TO	TABLE public.room TO
warehouse clerk;	warehouse clerk;
GRANT SELECT ON TABLE	GRANT SELECT, USAGE ON SEQUENCE
<pre>public.invoice_details_view TO</pre>	<pre>public.room_room_id_seq TO</pre>
warehouse_owner;	warehouse_owner;
GRANT SELECT, UPDATE ON TABLE	GRANT SELECT, USAGE ON SEQUENCE
<pre>public.invoice_details_view TO warehouse clerk;</pre>	<pre>public.room_room_id_seq</pre>
GRANT SELECT, INSERT, DELETE, UPDATE ON	GRANT SELECT, USAGE ON SEQUENCE
TABLE public.invoice_details_view TO	<pre>public.room_warehouseid_seq TO</pre>
<pre>warehouse_manager;</pre>	<pre>warehouse_owner;</pre>
GRANT SELECT, USAGE ON SEQUENCE	GRANT SELECT, INSERT, DELETE, UPDATE ON
public.invoice_employee_granted_acce	TABLE public.shelf TO
<pre>ss_seq TO warehouse_owner; GRANT SELECT,USAGE ON SEQUENCE</pre>	<pre>warehouse_owner; GRANT SELECT,INSERT,DELETE,UPDATE ON</pre>
public.invoice employee invoiceid se	TABLE public.shelf TO
q TO warehouse owner;	warehouse clerk;
GRANT SELECT, USAGE ON SEQUENCE	GRANT SELECT, USAGE ON SEQUENCE
<pre>public.invoice_employee_responsible_</pre>	<pre>public.shelf_rackid_seq T0</pre>
seq TO warehouse_owner;	warehouse_owner;
GRANT SELECT, USAGE ON SEQUENCE public.invoice invoice id seq TO	GRANT SELECT, USAGE ON SEQUENCE public. shelf shelf id seq TO
warehouse manager;	warehouse owner;
GRANT SELECT, USAGE ON SEQUENCE	GRANT SELECT, USAGE ON SEQUENCE
<pre>public.invoice_invoice_id_seq TO</pre>	<pre>public.shelf_shelf_id_seq TO</pre>
<pre>warehouse_owner;</pre>	<pre>warehouse_clerk;</pre>
GRANT SELECT, INSERT ON TABLE	GRANT SELECT, INSERT, DELETE, UPDATE ON
public.log_table TO warehouse_owner;	TABLE public.warehouse TO
GRANT SELECT, INSERT ON TABLE public.log table TO warehouse clerk;	<pre>warehouse_owner; GRANT SELECT,INSERT,DELETE,UPDATE ON</pre>
GRANT SELECT, INSERT ON TABLE	TABLE public.warehouse TO
public.log_table TO	warehouse_clerk;
warehouse_manager;	GRANT SELECT, INSERT, DELETE, UPDATE ON
	TABLE public.warehouse_details_view
	TO warehouse_owner;

Изм.	Лист	№ докум.	Подпись	Дата

GRANT SELECT, INSERT, UPDATE ON TABLE public.warehouse_details_view TO ALTER DEFAULT PRIVILEGES FOR ROLE warehouse_manager; postgres IN SCHEMA public GRANT GRANT SELECT, USAGE ON SEQUENCE SELECT ON TABLES TO public.warehouse_warehouse_id_seq TO warehouse_owner; warehouse_owner;

warehouse_clerk; postgres IN SCHEMA public GRANT ALL GRANT SELECT ON TABLE ON FUNCTIONS TO warehouse_owner; public.warehouse_details_view TO ALTER DEFAULT PRIVILEGES FOR ROLE

Изм.	Лист	№ докум.	Подпись	Дата

Приложение Е Отзыв руководителя



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

ОТЗЫВ руководителя на курсовую работу

Волковой Эмили (Ф.И.О. студен	
Проектирование и реализация (
(наименование теми	
представленный к защите по на	
02.03.03 Математическ	<u> </u>
администрирование инфор	
(код и наименование направления	*
Программное обеспечение	
<u>И автоматизирован</u> (наименования профиля	
ст. преподаватель (должность)	/ <u>Новиков С. П.</u> (подпись) (Ф.И.О) «»20

Изм.	Лист	№ докум.	Подпись	Дата

KP.350000.000