Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИУК «Информатика и управление»**

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему:

Разработка системы электронного документа оборота

по дисциплине *Компьютерные сети*

Студент гр. ИУК4-72Б _____ (Калашников А.С.)
(подпись) (Ф.И.О.)

Руководитель _____ (Красавин Е.В.)
(подпись) (Ф.И.О.)

Оценка руководителя _____ баллов _____
30-50 (дата)

Оценка защиты _____ баллов _____
30-50 (дата)

Оценка проекта _____ баллов _____
(оценка по пятибалльной шкале)

Комиссия: _____ (Красавин Е.В.)
(подпись) (Ф.И.О.)

_____ (Гагарин Ю.Е.)
(подпись) (Ф.И.О.)

_____ (Белов Ю.С.)
(подпись) (Ф.И.О.)

Калуга, 2023

Калужский филиал
федерального государственного бюджетного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
_____ (_____)
«____» _____ 2023г.

ЗАДАНИЕ
на выполнение курсового проекта

по дисциплине **Компьютерные сети**

Студент Калашниокв А.С., ИУК4-72Б
(фамилия, инициалы, индекс группы)

Руководитель Красавин Е.В.
(фамилия, инициалы)

График выполнения работы: 25% к 6 нед., 50% к 9 нед., 75% к 11 нед., 100% к 14 нед.

1. Тема курсового проекта
Разработка системы электронного документа оборота

2. Техническое задание
Спроектировать систему электронного документа оборота

3. Оформление курсового проекта
3.1. Расчетно-пояснительная записка на _ листе формата А4.
3.2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.)
1. *Архитектура приложения*
2. *Структура БД*
3. *Демонстрационный чертеж*

Дата выдачи задания «*09* » *сентября 2023*г.

Руководитель курсовой работы _____ / Красавин Е.В.
(подпись) (Ф.И.О.)

Задание получил _____ / Калашников А.С. / «*09*» *сентября* 20*23* г.
(подпись) (Ф.И.О.)

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Наше время называют «информационным веком». Это название возникло потому, что самым важным, ценным и необходимым ресурсом является информация. Обладание информацией и умение своевременно, быстро и четко донести необходимую информацию до клиентов – ключ к успеху в реалиях современного бизнеса. В настоящий момент именно Интернет, способен оперативно и массово передавать текст, звук, изображения и даже видео-ролики, т.е. практически любую информацию. С точки зрения пользователя, Интернет – это огромный информационный ресурс, в котором можно найти все, что угодно: от прогноза погоды до личных предпочтений голливудских кинозвезд. Аудитория сети Интернет растет с каждым годом, Интернет – самое быстроразвивающееся средство передачи информации за все историю человечества. Пользователи сети интернет являются целью особого интереса для рекламодателей, потому что факт остается фактом – среди пользователей сети куча потенциальных клиентов. Сфера услуг являются одной из самых перспективных сфер экономики и экономической деятельности. Она охватывает широкий спектр экономической деятельности и по своему функциональному назначению не является единым комплексом. В процессе коммерциализации российской экономики и при Российских реалиях: большие расстояния, острая нехватка времени; особую значимость приобретает такая важная форма реализации товаров и услуг, как торговля и реклама через сеть Интернет. Существует огромное количество способов коммерческого подхода к сети интернет. В сети можно рекламировать услуги, продавать товары либо представлять потребителю и то и другое. В современном бизнесе многое зависит от самопрезентации компании, ее позиционирования на рынке оказываемых услуг и способности искать новых клиентов и рынки сбыта. Одним из инструментов, как имиджевых, так и маркетинговых, является наличие своего сайта в сети Интернет. С точки зрения бизнеса. Интернет – это современная рекламная площадка, позволяющая обеспечить приток клиентов. Web-приложение превращает компанию в современный бизнес. Web-приложение компании является важнейшим источником

информации для потенциальных клиентов и людей, чье мнение является общественно значимым.

Целью учебной практики является реализация веб-приложения на базе odoo для быстрого взаимодействия сотрудников компании с товаром.

Для достижения поставленной цели ставятся следующие задачи:

- Проанализировать предметную область;
- Создать и интегрировать репозиторий Git;
- Развернуть сервер на ПК через odoo;
- Создать собственный пустой модуль;
- Реализовать требуемый функционал.

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1. Введение

1.1.1. Наименование системы

Настоящее Техническое задание определяет требования и порядок создания программного продукта «Веб-приложение для закупкуи и продажи товаров».

1.1.2. Основания для разработки

Разработка приложения для контроля склада, покупок и продаж может быть полезной для компаний, занимающихся торговлей и логистикой, а также для любой компании, которая имеет потребность эффективно управлять своими запасами.

Такая задача позволит эффективно управлять своими запасами, позволяя им оптимизировать процессы заказа и хранения товаров. Система может контролировать уровень запасов, предупреждать о необходимости пополнения запасов, автоматически формировать заказы и учитывать изменения в динамике спроса.

Также система может обеспечивать более эффективное управление заказами, отслеживание доставки и сроков выполнения заказов, автоматическое формирование счетов на оплату и уведомления клиентов о статусе их заказов.

Приложение предоставит точные и надежные данные и различные инструменты для анализа и отчетности, которые помогают компании принимать стратегические решения в отношении закупок, продаж и управления запасами. Система может автоматически обновлять информацию при операциях и предотвращать ошибки в данных и уменьшать риски потерь или несоответствия запасов и продаж.

1.1.3. Исполнитель

Исполнителем проекта является студент Калужского филиала МГТУ им. Н. Э. Баумана, факультета ИУК, группы ИУК4-62Б, Калашников Артем Сергеевич.

1.1.4. Краткая характеристика области применения

Разрабатываемая система предназначена для использования сотрудниками производства

1.2. Назначение и цель разработки системы

1.2.1. Назначение системы

Разрабатываемая система должна предоставлять возможность пользователям взаимодействовать с товарами и услугами. Получать доступы к имеющимся продуктам.

1.2.2. Цели создания системы

Целью создания системы является ускорение взаимодействия пользователй с продукцией компании и облегчение данного процесса

1.3. Исследование предметной области задачи и постановка задачи

Odoo представляет собой интегрированную открытую платформу для управления бизнес-процессами. Она основана на языке Python и базе данных PostgreSQL. Одна из основных особенностей Odoo - это модульная структура,

позволяющая разработчикам создавать и устанавливать различные модули для расширения функциональности системы.

Основные компоненты Odoo включают в себя:

1. Модули: Odoo состоит из различных модулей, которые предоставляют функциональность для управления различными аспектами бизнеса, такими как учет, продажи, закупки, склад, производство и другие. Модули можно устанавливать и настраивать в соответствии с потребностями компании.

2. Веб-интерфейс: Odoo предоставляет удобный веб-интерфейс для доступа и управления данными и модулями. Веб-интерфейс отличается интуитивно понятным дизайном и легкостью использования.

3. Управление бизнес-процессами: Odoo предлагает инструменты для ав томатизации и управления бизнес-процессами. Это включает в себя создание и настройку рабочих потоков, определение прав доступа и ролей пользователей, управление задачами и уведомлениями, а также отчетность и аналитику.

4. Интеграция: Odoo предлагает мощные возможности интеграции, позволяющие связать систему с другими внешними приложениями и сервисами. Интеграция может быть осуществлена через API и Web-сервисы, что обеспечивает обмен данными и автоматизацию процессов с другими системами.

5. Многовалютность и многолокализация: Odoo поддерживает работу с различными валютами, а также учитывает требования локализации для различных стран и регионов. Модули локализации содержат настройки для различных аспектов, таких как налоги, учетные отчеты, форматирование данных и т.д.

Общая гибкость, доступность и возможность расширения делают Odoo популярной и практической системой для автоматизации бизнес-процессов во многих отр раслях и секторах деятельности. Odoo широко используется в различных отраслях, таких как производство, торговля, услуги, розничная торговля, логистика и многие другие.

Одной из ключевых особенностей Odoo является его гибкость и настраиваемость. Благодаря модульной структуре, вы можете выбирать те

модули, которые соответствуют вашим потребностям, и настраивать их для ваших конкретных бизнес-процессов. Кроме того, вы можете разрабатывать собственные модули, добавлять новые функции и интегрировать сторонние приложения для создания полностью индивидуализированной системы управления.

Odoo предоставляет разнообразные функции, включая управление продажами и закупками, финансовый учет, управление складом и производством, управление проектами и задачами, CRM, а также много других инструментов для эффективного управления бизнесом.

Одним из главных преимуществ использования Odoo является то, что это открытая платформа с открытым исходным кодом. Это означает, что вы имеете доступ к исходному коду системы, что позволяет вам изменять и настраивать ее под свои потребности и требования вашего бизнеса. Вы также можете вступить в различные сообщества и форумы, где можно задавать вопросы, делиться своими находками и получать помощь от опытных пользователей и разработчиков Odoo.

В целом, Odoo предлагает мощные инструменты и функции для автоматизации и управления бизнес-процессами. Он является гибким и настраиваемым решением, которое может адаптироваться под различные отрасли и требования бизнеса. Благодаря его разносторонним возможностям и гибкости, Odoo стал популярным выбором для компаний разных масштабов, от небольших предприятий до крупных корпораций.

1.4. Актуальность решаемой проблемы и возможные области применения данной разработки

Разработка сайтов является актуальным занятием. Актуальность создания сайта состоит в том, что если вы хотите донести информацию максимально быстро до огромного количества людей, то лучше, чем с помощью собственного сайта сделать это не получится никак. Веб-ресурс позволяет

представить информацию о компании и ее товарах или услугах сжато и одновременно полноценно. Также сайт может сообщать о новостях, об изменениях, содержать отзывы благодарных клиентов.

Актуальность разработки сайта объясняется следующими факторами:

- Быстрота подачи информации широкому кругу лиц;
- Улучшение имиджа компании и повышение ее популярности;
- Возможность организовать обратную связь с клиентами;
- Увеличение трафика.

1.5. Календарный план

Таблица 1.

Календарный план

Дата	Задание
3 июля	Развернул сервер Odoo у себя на ПК через Docker
4 июля	Создание собственного пустого модуля с названием “test_module”
5 июля	Разобрался как проводить установку модулей через веб-интерфейс
6 июля	Настраивал автоматическое обновление модулей
7-10 июля	Создал собственную модель с названием “test.model” с различными полями
10 июля	Создал отдельный wizard с полем

11-14 июля	Добавил в представление формы (form) кнопку “Создать и изменить”

2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

2.1. Обоснование выбора инструментов и платформы для разработки клиентской и серверной частей приложения

В качестве средства ведения репозитория Git был выбран GitHub, так как он удобен и наиболее распространен.

Python

Выбор Python как средства для реализации проекта можно обосновать несколькими факторами:

1. Простота и удобство использования: Python имеет простой и читаемый синтаксис, который делает его легким для изучения и использования. Это позволяет разработчикам быстрее создавать и поддерживать код.

2. Большое сообщество разработчиков: Python обладает активным сообществом разработчиков, что означает наличие обширной поддержки, документации, библиотек и фреймворков. Это делает его удобным средством выбора для реализации проекта, так как возможностей и решений уже разработано и протестировано множество.

3. Портативность: Python является переносимы языком программирования, что означает, что код, написанный на Python, может быть запущен на различных платформах и операционных системах без необходимости внесения больших изменений. Это упрощает развертывание и обеспечивает гибкость в использовании проекта на различных окружениях.

4. Интеграция с другими языками: Python также обладает хорошей интеграцией с другими языками программирования, что может быть полезным при взаимодействии с существующими системами или библиотеками, написанными на других языках, таких как C/C++.

5. Масштабируемость: Python предлагает множество инструментов и подходов для разработки масштабируемых приложений. Возможность создания распределенных систем, параллельной обработки и управлением большим объемом данных делает Python привлекательным для разработки проектов любого масштаба.

Python сочетает в себе простоту, гибкость и производительность, что позволяет разработчикам эффективно создавать приложения и решать задачи разного уровня сложности. Большое количество доступных библиотек и фреймворков расширяет возможности Python и упрощает разработку специализированных решений.

Обоснование выбора СУБД

Разрабатываемая система предполагает хранение и обработку данных. Наилучшим способом работы с информацией, учитывая эти цели, является использование базы данных. Для взаимодействия с базами данных используются системы управления базами данных (СУБД).

Несмотря на то, что все системы управления базами данных выполняют одну и ту же основную задачу: дают возможность пользователям создавать, редактировать и получать доступ к информации, хранящейся в базах данных, сам процесс выполнения этой задачи варьируется в широких пределах. Кроме того, функции и возможности каждой СУБД могут существенно отличаться.

Существует шесть основных типов баз данных: иерархическая, сетевая, реляционная, объектно-ориентированная, объектно-реляционная, функциональная.

В приложениях, работающих с СУБД, которые следуют иерархической или сетевой модели, структура базы данных «зашита» в само приложение. Это означает, что приложение зависит от определенной физической реализации базы данных. При добавлении в базу данных нового атрибута, вне зависимости от его

использования приложением, придется изменять принцип работы приложения, что препятствует непрерывной работе приложения.

Функциональные базы данных используются для решения аналитических задач, таких как финансовое моделирование и управление производительностью. Включает в себя многомерное иерархическое объединение.

Реляционная модель данных позволяет представлять информацию о предметной области с помощью взаимосвязанных таблиц. В реляционных базах данных вся информация сведена в таблицы, строки и столбцы, которые называются записями и полями соответственно. Записи в таблицах не повторяются, их уникальность обеспечивается первичным ключом, содержащим набор полей, однозначно определяющих запись.

Объектно-реляционные базы становятся популярны и являются реляционными базами данных с объектно-ориентированными подходами, что является удобным инструментом. Но такие базы переносят часть логики с сервера в базу данных, что противоречит принципу независимости данных от приложения, что не даёт разработчикам отказаться от реляционных моделей.

Для системы было решено использовать либо реляционную, либо объектно-реляционную базу данных. Это обосновано рядом преимуществ, среди которых: простота и наглядность по сравнению с другими моделями, удобство физической реализации, применение строгих правил при проектировании баз данных, независимость данных, их логическая связь, широкая техническая поддержка, наличие стандартов и документации.

Для разработки базы данных было необходимо выбрать наиболее подходящую СУБД для поставленной задачи. В силу распространенности, для рассмотрения были выбраны: PostgreSQL, SQLite, Firebird, MySQL и Oracle.

PostgreSQL

PostgreSQL – свободно распространяемая СУБД, относится к объектно-реляционному типу. Существует в реализациях для множества UNIX-подобных

платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного и/или реляционного подхода к базам данных. Например, полная поддержка надежных транзакций, то есть атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability – ACID). Благодаря мощным технологиям PostgreSQL очень производительна. PostgreSQL очень легко расширять хранимыми процедурами.

Хотя PostgreSQL и не может похвастаться большой популярностью в отличии от MySQL, существует довольно большое число приложений, облегчающих работу с PostgreSQL, несмотря на всю мощность функционала.

Преимущества:

1. Полностью свободная лицензия;
2. Полная SQL-совместимость;
3. Большое сообщество. Благодаря этому можно легко найти ответ на возникший при работе с СУБД вопрос;
4. Расширяемость. PostgreSQL можно программно расширить за счёт хранимых процедур;
5. Объектно-ориентированность. PostgreSQL не только реляционная, но и объектно-ориентированная СУБД.

Недостатки:

1. В простых операциях чтения PostgreSQL может проигрывать в скорости своим соперникам, например, MySQL;
2. Несмотря на большое сообщество, данную СУБД трудно назвать очень популярной.

За свою производительность, расширяемость и удобство работы для разработки базы данных в данной курсовой работе была выбрана СУБД PostgreSQL.

SQLite

SQLite – встраиваемая компактная реляционная база данных. Особенность SQLite в том, что она не использует парадигму клиент-сервер, поэтому её и называют «встраиваемой». Движок SQLite не отдельно работающий процесс, а библиотека, с которой программа компонуется и движок становится составной частью программы.

Преимущества:

1. Полностью свободная лицензия;
2. Файловая структура. Вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины;
3. Кроссплатформенность;
4. Безопасность. БД хранится в одном файле, права доступа к которому можно контролировать стандартными средствами ОС.

Недостатки:

1. Встраиваемость. Подходит только для создания локальной базы данных.
2. Отсутствие системы пользователей. Более крупные СУБД включают в свой состав системы управления правами доступа пользователей. Обычно применения этой функции не так критично, так как эта СУБД используется в небольших приложениях.
3. Отсутствие возможности увеличения производительности. Исходя из проектирования, довольно сложно выжать что-то более производительное из этой СУБД.

MySQL

MySQL – одна из самых популярных баз данных для веб-приложений. Она включает в себя бесплатный пакет программ, однако новые версии выходят постоянно, расширяя функционал и улучшая безопасность. Существуют специальные платные версии, предназначенные для коммерческого

использования. В бесплатной версии наибольший упор делается на скорость и надежность, а не на полноту функционала, который может стать и достоинством, и недостатком - в зависимости от области внедрения

В силу того, что MySQL является серверной СУБД, приложения для доступа к данным, в отличии от SQLite работают со службами MySQL. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Преимущества:

1. Простота. MySQL легко устанавливается и проста в настройке и использовании. Существует много сторонних инструментов, облегчающих работу с базами данных;
2. Безопасность. Имеет много функций, обеспечивающих безопасность и поддерживаемых по умолчанию;
3. Масштабируемость. MySQL легко работает с большими объемами данных и просто масштабируется;
4. Скорость. Упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность;

Недостатки:

1. Ограничения. По задумке в MySQL заложены некоторые ограничения функционала, которые иногда необходимы в особо требовательных приложениях;
2. Надёжность. Из-за некоторых способов обработки данных MySQL (связи, транзакции, аудиты) иногда уступает другим СУБД по надежности.

FireBird

FireBird – свободно распространяемая СУБД. Поддерживает стандарты ANSI в синтаксисе языка SQL и позволяет работать на многих операционных

системах. Firebird используется в различных промышленных системах (складские и хозяйственные, финансовый и государственный сектора).

Преимущества:

1. Многоверсионная архитектура (параллельная обработка оперативных и аналитических запросов: читающие пользователи не блокируют пишущих);
2. Соответствие требованиям ACID.

Недостатки:

1. Отсутствие кэша итогов запросов, полнотекстовых индексов;
2. Значительное падение производительности при росте внутренней фрагментации базы.

Oracle

Oracle Database или Oracle RDBMS – объектно-реляционная система управления базами данных компании Oracle. Современная СУБД Oracle – это мощный программный комплекс, позволяющий создавать приложения любой степени сложности. Ядром этого комплекса является база данных, хранящая информацию, количество которой за счет предоставляемых средств масштабирования практически безгранично. С высокой эффективностью работать с этой информацией одновременно может практически любое количество пользователей (при наличии достаточных аппаратных ресурсов), не проявляя тенденции к снижению производительности системы при резком увеличении их числа.

Преимущества:

1. Не требует больших объемов памяти для кеша. При этом память, не задействованная для кеширования файловой системы, может быть сконфигурирована для Oracle memory.;
2. Масштабируемость;
3. Высокая скорость выполнения запросов.

Недостатки:

1. Коммерческая направленность продукта, высокая стоимость;
2. Необходимость дорогостоящего аппаратного обеспечения;
3. Миграция из устаревших файловых систем в ASM может быть проблемой и часто требует отключения системы;

Выводы

Таким образом, исходя из требований к заявленному приложению был проведен анализ нескольких инструментов для разработки, что послужило к формированию стека следующих технологий:

— для разработки клиентской части приложения был выбран Odoo и поэтому был выбран Python

— для разработки серверной части приложения были выбраны следующие инструменты: СУБД PostgreSQL.

Средой разработки выступила VisualStudio Code. Так как данная среда поддерживает плагины, необходимые для работы, бесплатна и быстра в использовании.

2.2. Принцип работы приложения

Odoo состоит из различных модулей, предоставляющих функциональность для управления различными бизнес-процессами. Модули могут включать в себя учет, продажи, CRM, закупки, складское хозяйство и другие. Каждый модуль предоставляет специализированную функциональность и инструменты для выполнения соответствующих бизнес-процессов. Например, модуль учета предлагает функции по учету финансовой информации, модуль продаж обеспечивает инструменты для управления продажами и заказами, а модуль складского хозяйства позволяет управлять складскими операциями (рис. 1).

Рис.1 Добавление модуля в системе

Odoo хранит различные данные, такие как клиенты, товары, заказы и другую информацию, в своей базе данных PostgreSQL. База данных служит центральным хранилищем данных для приложения (рис. 2).

S00454

Заказчик	ТЭ МСК	**Дата заказа**	16.08.2023 15:12:31

№		Продукт	Полное название
1		[57814] 1ПКВТ-10-300/400(Б)	Муфта кабельная концевая 1ПКВТ-10-300/400(Б) (КВТ)
2		[57810] 1ПКВТ-10-70/120(Б)	Муфта кабельная концевая 1ПКВТ-10-70/120(Б) (КВТ)

Рис. 2 Различные поля

Также с данными есть возможность взаимодействовать и добавлять новые значения (рис.3).

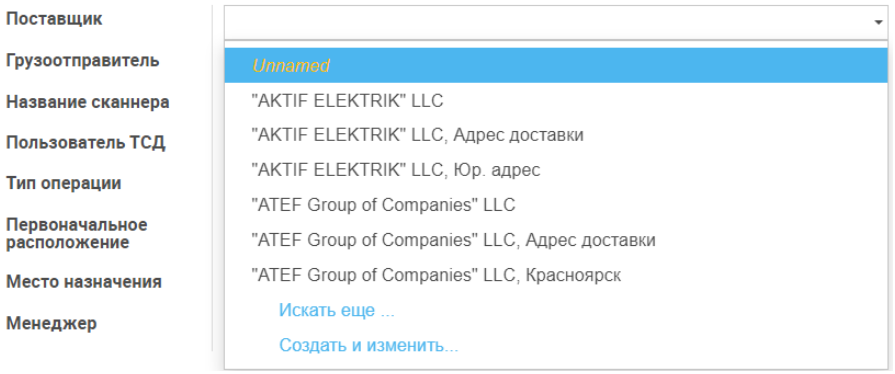

Рис. 3 Работа со значениями

2.3. Проектирование базы данных

База данных состоит из следующих таблиц:

1. sale_order
2. sale_order_line

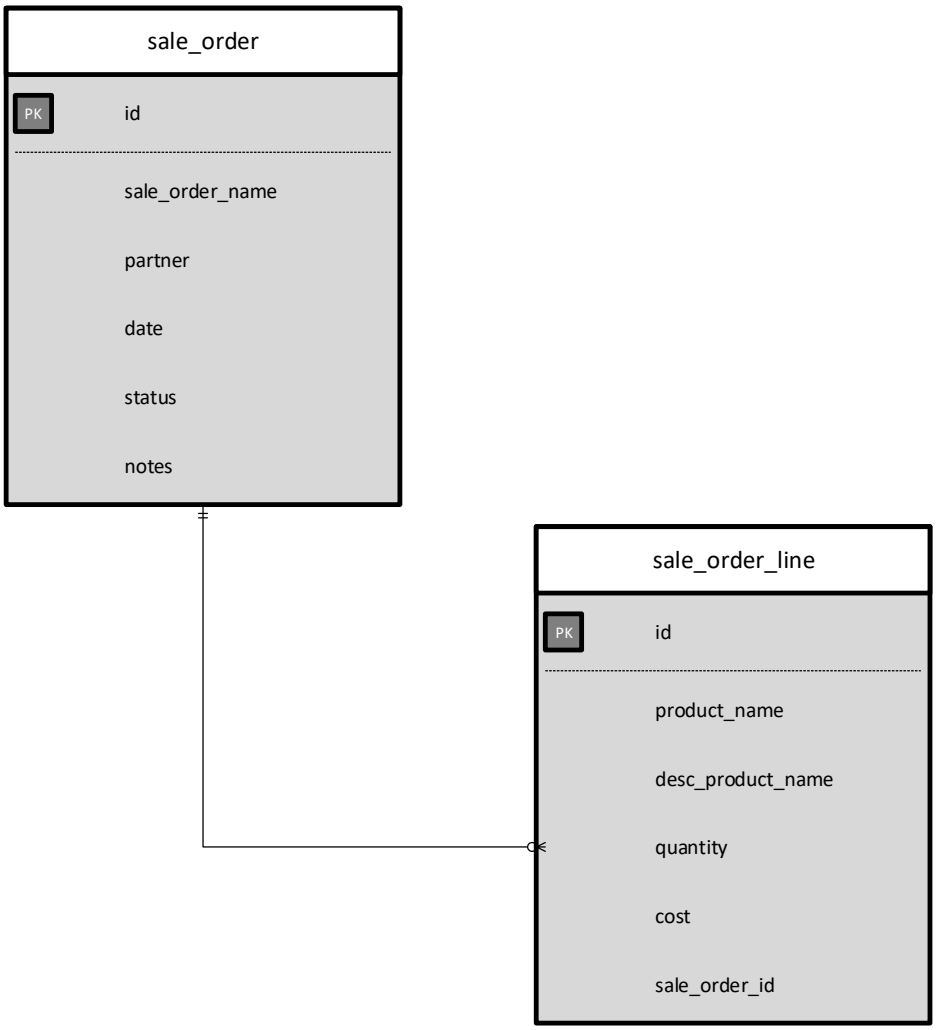

Рис. 5 Структура базы данных

2.4. Описание программных модулей

Модули серверной части приложения имеют следующую структуру:

1. Таблица sale_order

- Поле id – уникальный идентификатор, реализованный путем автоинкрементирования при создании

- Поле sale_order_name – название заказа

- Поле partner – наименовенае заказчика

- Поле date – дата заказа

- Поле status – статус заказа

- Поле notes – комментарии к заказу

1. Таблица sale_order_line

- Поле id – уникальный идентификатор, реализованный путем автоинкрементирования при создании

- Поле product_name – название заказа

- Поле desc_product_name – наименовенае заказчика

- Поле quantity – дата заказа

- Поле cost – статус заказа

- Поле sale_order_id – комментарии к заказу

2.5. Обработка исключений

Для обработки возможных ошибок были выявлены точки возможного возникновения исключительных ситуаций.

И в таких ситуациях используя конструкцию try except были добавленны сообщения, уведомляющие пользоваетля об ошибке (рис. 6).

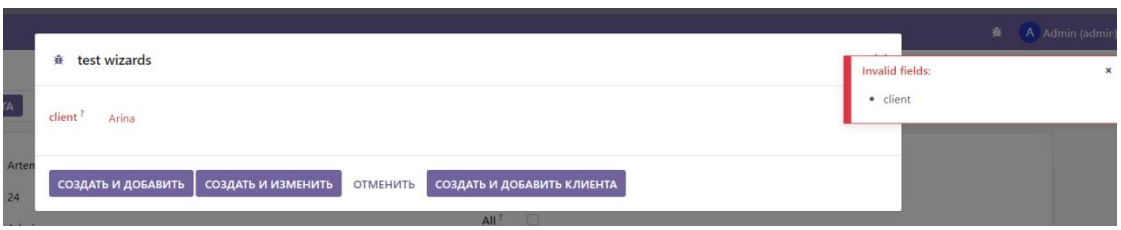

Рис. 6 Сообщение об ошибке

ЗАКЛЮЧЕНИЕ

Поставленная цель проектно-технологического практикума– была успешно достигнута. Для приложения разработаны серверная и клиентская части, определена логика взаимодействия между ними. Был разработан графический интерфейс, изучены различные способы взаимодействия пользователя с ним и возникающие в их результате исключительные ситуации.

В ходе работы над проектом были получены практические навыки создание WEB-приложений, ведения Git-репозитория на сервисе github, правильного разворота сервера odoo у себя на ПК используя Docker. Также были получены базовые знания работы с odoo и с xml.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Основная литература

1. Моделирование информационных ресурсов [Электронный ресурс]: учебно-методический/ Составитель Огнев Э.Н. - Кемерово: Кемеровский государственный университет культуры и искусств, 2013. - 36 с.: ил., табл. - URL: http://biblioclub.ru/index.php?page=book&id=274218

2. Вайнштейн, М. З. Основы научных исследований [Электронный ресурс: учебное пособие / М. З. Вайнштейн, В. М. Вайнштейн, О. В. Кононова. — Йошкар-Ола: Марийский государственный технический университет, Поволжский государственный технологический университет, ЭБС АСВ, 2011. — 216 с. — Режим доступа: http://www.iprbookshop.ru/22586.html

3. Гданский Н.И. Основы теории и алгоритмы на графах: учебное пособие. – М.: Инфра-М, 2020. – 206 с.

4. Дронов, В.А. Python 3 и PyQt 5. Разработка приложений. – СПб.: БХВ-Петербург, 2016. – 832 с.

5. Коваленко, Ю.В. Информационно-поисковые системы [Электронный ресурс]: учебно-методическое пособие / Ю.В. Коваленко, Т.А. Сергиенко. — Омск: Омская юридическая академия, 2017. — 38 с.— Режим доступа: http://www.iprbookshop.ru/66817.html

6. Маюрникова, Л. А. Основы научных исследований в научно-технической сфере [Электронный ресурс]: учебно-методическое пособие / Л. А. Маюрникова, С. В. Новосёлов. — Кемерово: Кемеровский технологический институт пищевой промышленности, 2009. — 123 с. — Режим доступа: http://www.iprbookshop.ru/14381.html

7. Мокий, М.С. Методология научных исследований [Текст]: учебник / М.С. Мокий, А.Л. Никифоров, В.С. Мокий. - М.: Юрайт, 2015. - 255 с.

8. Рафгарден, Т. Совершенный алгоритм. Графовые алгоритмы и структуры данных. – СПб.: Питер, 2019 – 256 с.

9. Рогов, В.А. Методика и практика технических экспериментов [Текст]: учеб.пособие / В.А. Рогов, А.В. Антонов, Г.Г. Поздняк. – М.: Академия, 2005. – 288 с.

10. Хайнеман, Дж. Алгоритмы. Справочник с примерами на C, C++, Java и Python, 2-е изд.: Пер. с англ. / Хайнеман, Дж., Поллис, Г., Селков, С. – СПб.: Альфа-книга, 2017. – 432 с.

11. Харари, Ф. Теория графов. – М.: Мир, 1973. – 300с.

12. Дронов, В. React 17. Разработка веб-приложений на JavaScript. – СПб.: БХВ-Петербург, 2022.

Дополнительная литература

13. Щербаков, А. Интернет-аналитика [Электронный ресурс]: поиск и оценка информации в web-ресурсах: практическое пособие / А. Щербаков. - М.: Книжный мир, 2012. - 78 с. - URL: http://biblioclub.ru/index.php?page=book&id=89693 .

14. Моделирование систем [Текст]: учебник для вузов / С.И. Дворецкий, Ю.Л. Муромцев, В.А. Погонин, А.Г. Схиртладзе. – М.: Академия, 2009. – 320 с.

15. Порсев, Е. Г. Организация и планирование экспериментов [Электронный ресурс]: учебное пособие / Е. Г. Порсев. — Новосибирск : Новосибирский государственный технический университет, 2010. — 155 с. — Режим доступа: http://www.iprbookshop.ru/45415.html

ПРИЛОЖЕНИЯ

models.py:

```
from odoo import api, models, fields

class TestModels(models.Model):
    _name = 'test.models'
    _description = 'Test models'

    test_name = fields.Char(string = 'Name', required=True)

    test_age = fields.Integer(string='Age')

    gender = fields.Selection([
        ('male','Male'),
        ('female','Female'),
    ], required=True, default='male',string='Gender')

    notes = fields.Text(string='Notes')

    clients=fields.One2many(comodel_name='test.model.lines',
inverse_name='test')

    @api.onchange('bool_all')
    def onchange_bool_all(self):
        values = {}
        if self.bool_all:
            values['one'] = self.bool_all
            values['two'] = self.bool_all
        if ((self.bool_all==False) and (self.one==True) and (self.two==True)):
            values['one'] = self.bool_all
            values['two'] = self.bool_all
        if ((self.bool_all==False) and (self.one==False) and (self.two==True)):
            values['one'] = self.bool_all
            values['two'] = True
        if ((self.bool_all==False) and (self.one==True) and (self.two==False)):
            values['one'] = True
            values['two'] = self.bool_all
        self.update(values)

    @api.onchange('two','one')
    def onchange_bool_two(self):
        values = {}
        if ((self.one==True) and (self.two==True)):
            values['bool_all'] = True
        if ((self.one==False) or (self.two==False)):
            values['bool_all'] = False
        self.update(values)
    def create_wizard(self):
        return{
            'view_type': 'form',
            'view_mode': 'form',
            'res_model': 'test.wizards',
            'target': 'new',
            'type': 'ir.actions.act_window',
            'context': {'test_models_id': self.id}
        }
```

models_list.py:

```
from odoo import api, models, fields
#Класс посредник между test.models и res.partner
class TestModelLines(models.Model):
    _name = 'test.model.lines'
    _description = 'Test models line'

    client_name = fields.Many2one(string='client name',
comodel_name='res.partner')
    email = fields.Char(string='Email', related ="client_name.email")
    test = fields.Many2one(comodel_name='test.models')
```

test_person.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<odoo>

 <!-- Treee-->
    <record id="test_tree" model="ir.ui.view">
        <field name="name">test.models.tree</field>
        <field name="model">test.models</field>
        <field name="arch" type="xml">
            <tree>
                <field name="test_name"/>
                <field name="test_age"/>
                <field name="create_name"/>
                <field name="responsible" attrs =
"{'invisible':[('create_name','=',False)]}"/>
                <field name="one"/>
                <field name="two"/>
                <field name="bool_all"/>
                <field name="image"/>
                <field name="clients"/>
            </tree>
        </field>
    </record>

 <!-- Form-->
    <record id="test_form" model="ir.ui.view">
        <field name="name">test.models.form</field>
        <field name="model">test.models</field>
        <field name="arch" type="xml">
            <form>
                <header>

                    <button
                    name="create_wizard"
                    string="Создать и добавить клиента"
                    type="object"
                    class="oe_highlight"/>
                </header>
               <sheet>
                    <group>


                        <group>
                            <field name="test_name"/>
                            <field name="test_age"/>
                            <field name="create_name"/>
                            <field name="responsible" attrs =
"{'invisible':[('create_name','=',False)]}"/>
                        </group>
                        <group>
```

```
                            <field name="one"/>
                            <field name="two"/>
                            <field name="bool_all"/>
                        </group>
                        <group attrs = "{'invisible':
                                ['|','&amp;',('create_name','!=',''),('one','=',
True),
                                '&amp;',('two','=',True),('responsible','!=','')
]}">
                            <field name="gender" attrs = "{'invisible':
                                ['|','&amp;',('create_name','!=',''),('one','=',
True),
                                '&amp;',('two','=',True),('responsible','!=','')
]}"/>
                            <field name="notes" attrs =
"{'invisible':[('create_name','=',False)]}"/>
                            <field name="image" attrs =
"{'invisible':[('create_name','=',False)]}"/>

                        </group>
                        <field name="clients"/>
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <record id="test_action" model="ir.actions.act_window">
        <field name="name">Tests</field>
        <field name="type">ir.actions.act_window</field>
        <field name="res_model">test.models</field>
        <field name="view_mode">tree,form</field>
        <field name="help" type="html"></field>
    </record>

<!-- Menu-->
    <menuitem id="test_root"
    name="Test"
    sequence="10"/>

    <menuitem id="test_test"
    name="Test2"
    parent="test_root"
    sequence="10"/>

    <menuitem id="test_test_root"
    name="Test5"
    parent="test_test"
    action="test_action"
    sequence="10"/>



</odoo>
```

test_person_list.tsx:

```
<?xml version="1.0" encoding="utf-8"?>

<odoo>
```

```
    <!-- Treee-->
    <record id="test_lines_tree" model="ir.ui.view">
        <field name="name">test.model.lines.tree</field>
        <field name="model">test.model.lines</field>
        <field name="arch" type="xml">
            <tree>
                <field name="client_name"/>
                <field name="email"/>
                <field name="test"/>

            </tree>
        </field>
    </record>

 <!-- Form-->
    <record id="test_lines_form" model="ir.ui.view">
        <field name="name">test.model.lines.form</field>
        <field name="model">test.model.lines</field>
        <field name="arch" type="xml">
            <form>
               <sheet>
                    <group>
                        <group>
                            <field name="client_name"/>
                            <field name="email"/>
                            <field name="test"/>
                        </group>
                    </group>
               </sheet>
            </form>
        </field>
    </record>

    <record id="test_line_action" model="ir.actions.act_window">
        <field name="name">Tests</field>
        <field name="type">ir.actions.act_window</field>
        <field name="res_model">test.model.lines</field>
        <field name="view_mode">tree,form</field>
        <field name="help" type="html"></field>
    </record>

</odoo>
```

test_wizard.py:

```
from odoo import models, fields, api, _
from odoo.exceptions import ValidationError
class TestWizards(models.TransientModel):
    _name='test.wizards'
    _description = 'Test wizards'

    partner_name = fields.Char(string='client', required=True)
    #  Функция добавить и создать
    def test_wizards(self):
        vals={
            'name': self.partner_name
        }

        self.check_func(vals)#Передает в функцию проверки
```

```
    #Функция добавить и изменить
    def test_wizards_change(self):
        vals={
            'name': self.partner_name
        }
        record_set_partner = self.check_func(vals) #возвращат объект
        return{
            'name': _('Change'),
            'type': 'ir.actions.act_window',
            'view_mode': 'form',
            'res_model':'res.partner',
            'res_id': record_set_partner.id, #id user
            'target': 'new',
        }

    @api.model
    def check_func(self, value):

        if self.env['res.partner'].search([('name','=',value['name'])]):
            raise ValidationError(_("Этот пользователь уже существует"))
        else:
            check_id=self.env['res.partner'].create(value)
            id_test_models_lines=self.env['test.model.lines'].create({'client_na
me':check_id.id})
            res =
self.env['test.models'].browse(self._context['test_models_id'])
            res.write({'clients':[(4,id_test_models_lines.id,0)]}) # you need to
add it as list

        return check_id
```

test_wizards.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
<record id="test_wizards_form" model="ir.ui.view">
    <field name="name">test.wizards.form</field>
    <field name="model">test.wizards</field>
     <field name="arch" type="xml">
        <form string="Create client">
            <group>
                <field name="partner_name"/>

            </group>
            <footer>
                <button
                    name="test_wizards"
                    string="Создать и добавить"
                    type="object"
                    class="btn-primary"/>
                <button
                    name="test_wizards_change"
                    string="Создать и изменить"
                    type="object"
                    class="btn-primary"/>
                <button
                    string="Отменить"
                    class="btn-secondary"
                    special="cancel"/>
            </footer>
        </form>
    </field>
```

```
</record>

<record id="create_test_wizard" model="ir.actions.act_window">
        <field name="name">test wizards</field>
        <field name="type">ir.actions.act_window</field>
        <field name="res_model">test.wizards</field>
        <field name="view_mode">form</field>
        <field name="view_id" ref="test_wizards_form"/>
        <field name="target">new</field>
    </record>
</odoo>
```