|  |  |
| --- | --- |
|  | **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»***\_\_\_\_\_\_

**О Т Ч Е Т**

**УЧЕБНАЯ ПРАКТИКА**

**«Проектно-технологическая практика»**

Студент гр. ИУК4-62Б \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_Калашников А.С.)

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_Пчелинцева Н.И.\_\_)

(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка практики \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_Гагарин Ю.Е.\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_Глебов С.А.\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_Пчелинцева Н.И.\_\_\_)

(подпись) (Ф.И.О.)

*Калуга, 2023Калужский 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 **\_\_ИУК4\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Гагарин Ю.Е.)

« 03 » июля 2023 г.

**ЗАДАНИЕ**

**на УЧЕБНУЮ ПРАКТИКУ, Проектно-технологическая практика**

За время прохождения практики студенту необходимо:

1. Развернуть сервер Odoo у себя на ПК через Docker, создать базу данных в веб интерфейсе. Результат выполнения - работающая ссылка localhost:8069 с домашней страницей Odoo.
2. Создать собственный пустой модуль с названием “test\_module”, расположить его в отдельной папке extra-addons.
   1. установить модуль в odoo через веб-интерфейс
   2. настроить автоматическое обновление модуля при перезапуске сервера odoo.
3. Создать собственную модель с названием “test.model” с различными полями
4. Создать отдельный wizard с полем
5. Добавить в представление формы (form) кнопку “Создать и изменить”
6. Подготовить отчет и защитить результаты практики.

Дата выдачи задания « 03 » июля 2023 г.

|  |  |  |
| --- | --- | --- |
| Руководитель практики |  | Пчелинцева Н.И. |
|  |  |  |
| Задание получил |  | Калашников А.С. |

**Оглавление**

[ВВЕДЕНИЕ 4](#_Toc113571262)

[1. ОБЩИЕ СВЕДЕНИЯ 6](#_Toc113571263)

[1.1. Исследование предметной области задачи и постановка задачи 6](#_Toc113571264)

[1.2. Обоснование выбора средства реализации 8](#_Toc113571265)

[1.3. Актуальность решаемой проблемы и возможные области применения данной разработки ации 9](#_Toc113571266)

[1.4. Определение входных и выходных характеристик 9](#_Toc113571267)

[1.5. Календарный план 10](#_Toc113571268)

[2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА 10](#_Toc113571269)

[2.1. Принцип работы приложения 10](#_Toc113571270)

[2.2. Проектирование иерархии классов 12](#_Toc113571271)

[2.3. Описание программных модулей 12](#_Toc113571272)

[2.4. Обработка исключений 13](#_Toc113571273)

[ЗАКЛЮЧЕНИЕ 16](#_Toc113571274)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc113571277)

[*Основная литература* 17](#_Toc113571278)

[*Дополнительная литература* 18](#_Toc113571279)

# 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. C 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. ОБЩИЕ СВЕДЕНИЯ

# 1.1. 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.

Общаясь с сообществом Odoo, вы также можете получить поддержку, обмен опытом и доступ к обширной библиотеке документации и ресурсов, которые помогут вам освоить и использовать Odoo более эффективно.

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.

# Обоснование выбора средства реализации

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

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.

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

# 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.

# Определение входных и выходных характеристик

Разрабатываемый программный продукт должен представлять собой приложение, которое будет взаимодействовать с базой данных, отображающее актуальную информацию

В проекте необходимо реализовать представление данных в виде таблиц в базе данных, которые будут отправляться клиенту для взаимодействия клиента с данными

# Календарный план

Таблица 1.

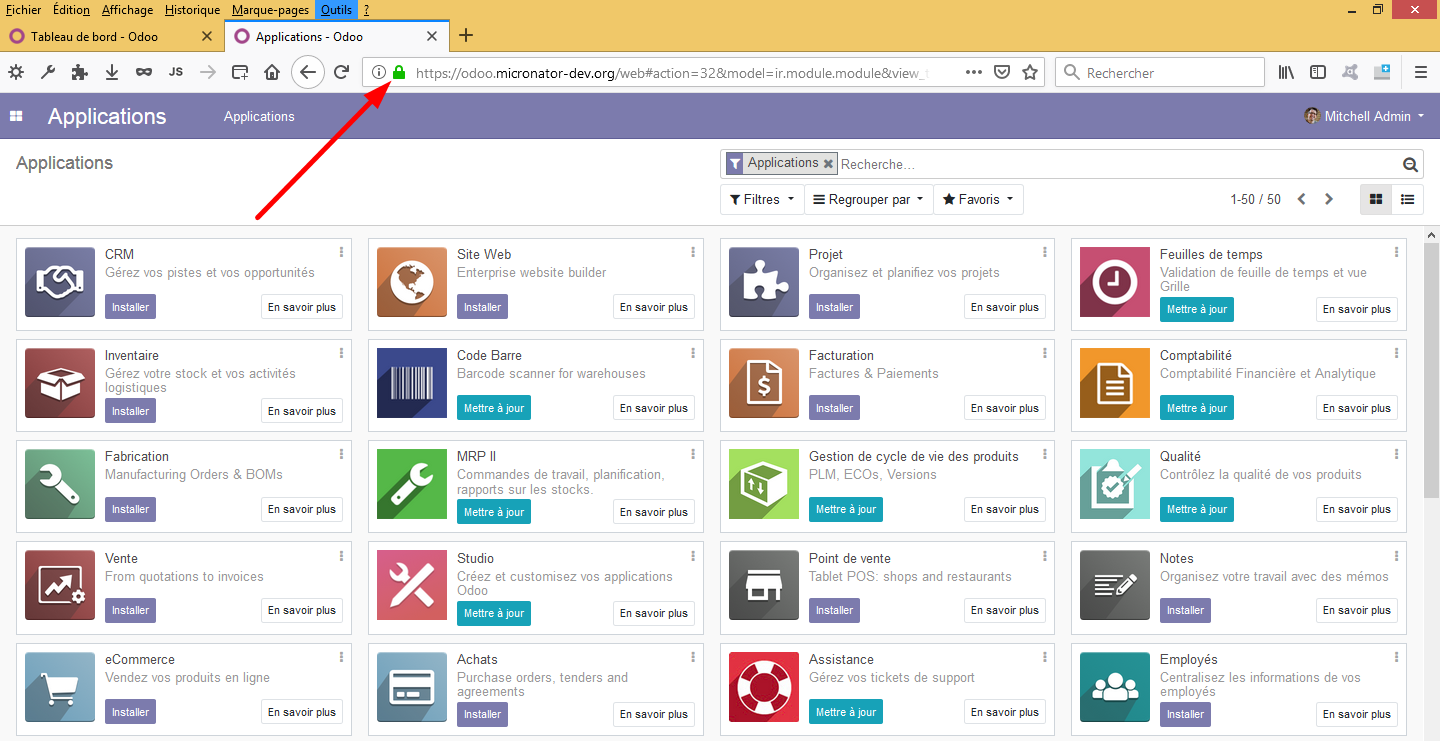
Календарный план

|  |  |
| --- | --- |
| **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

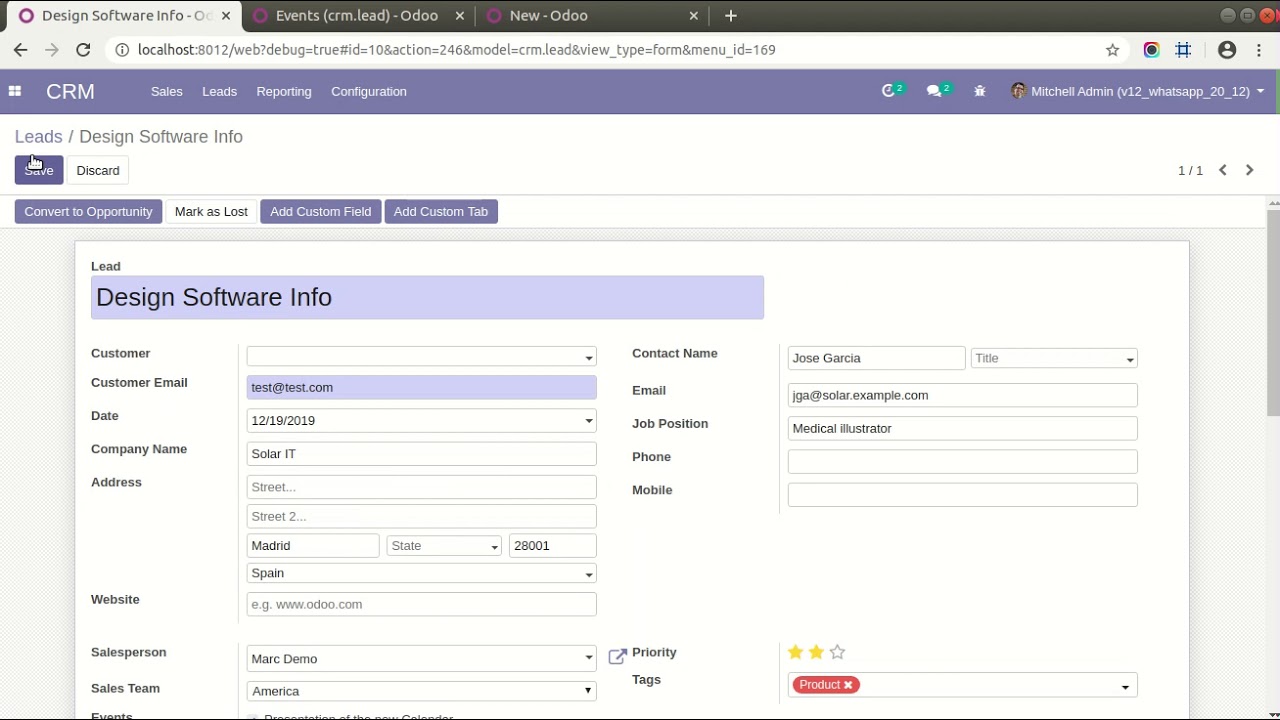
# 2.1. Принцип работы приложения

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).



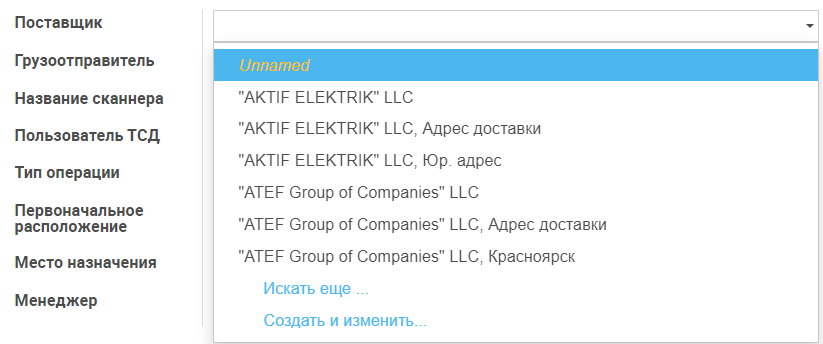
**Рис.1 Добавление модуля в системе**

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).



**Рис. 2 Различные поля**

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



**Рис. 3 Работа со значениями**

**2.2. Проектирование базы данных**

База данных состоит из следующих таблиц:

1. test\_models
2. test\_models\_lines



**Рис. 5** Структура базы данных

**2.3. Описание программных модулей**

Модули серверной части приложения имеют следующую структуру:

1. Таблица Card – сущность карточки

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

- Поле url – ссылка на изображение карточки

- Поле alt – строка для альтернативного вывода изображения

- Поле date – строка с датой публикации карточки

2. Функциональный JSX-компонент Card – реализация графического представления карточки для отображения пользователю.

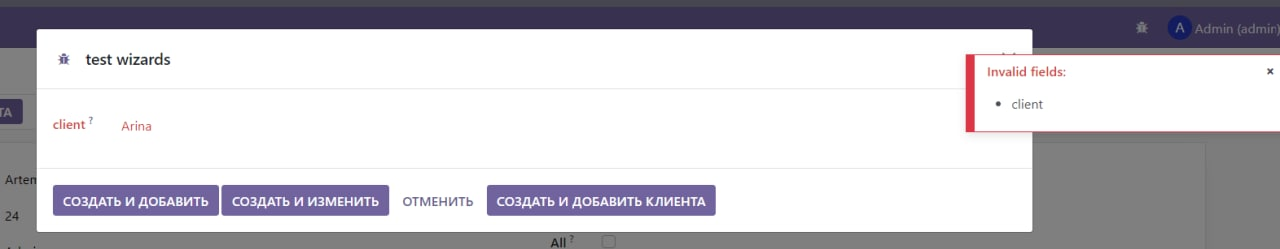
3. Функциональный JSX-компонент LoadButton – реалезация графического представления кнопки загрузки для отображения пользователю.

4. Функциональный JSX-компонент Home – реализация главной страницы с представлением списка из компонентов Card и выводом компонента LoadButton

**Рис. 7 –** структура разметки клиентского приложения

**2.4. Обработка исключений**

Для обработки возможных ошибок были выявлены точки возможного возникновения исключительных ситуаций.   
 При работе с библиотекой для асинхронных запросов к серверу “axios” использовался метод axios.get(), возвращающий особый тип данных – промис (Promise).



Интерфейс Promise (промис) представляет собой обёртку для значения, неизвестного на момент создания промиса. Он позволяет обрабатывать результаты асинхронных операций так, как если бы они были синхронными: вместо конечного результата асинхронного метода возвращается своего рода обещание (дословный перевод слова "промис") получить результат в некоторый момент в будущем.

У промисов имеются потребители: .then()/.catch()/.finally(), которые срабатывают практически аналогично конструкции try/catch/finally.   
 Во время асинхронного запроса могут возникнуть какие-либо ошибки (сервер недоступен, некорректные параметры запроса и т.д.), для обработки этих ошибок используется потребитель .catch(), в котором сообщение об ошибке выводится в виде алерта на экран пользователя.

**Рис. 8** – обработка исключения и демонстрация компонента Loader

Потребитель .finally() срабатывает аналогично блоку finally в конструкциях try/catch/finally и выполняется в любом случае после выполнения предыдущих блоков. Таким образом, после нажатия кнопки «ОК» (завершения блока catch) пользователю перестанет отображаться компонент Loader, сообщая, что никакая загрузка с сервера больше не идет.

**Рис. 9** – результат потребителя .finally() при возникновении исключения

В случае же, если никаких ошибок не возникло и промис вернул разрешение (resolve), произойдет замена пустого состояния с карточками ([cards]) на состояние, содержащее в себе объекты-карточки. Действия, выполняемые при успешном разрешении промиса выполняются потребителем .then().

**Рис. 10** – результат потребителя .then()

Также, при успешном разрешении промиса, все равно сработает блок .finally(), который уберет компонент Loader с экрана.

При нажатии на кнопку «загрузить еще» в хуке useEffect будет еще раз вызван метод axios.get() с такими же потребителями, что позволит отображать компонент Loader, пока происходит загрузка и выдать ошибку в любой момент обращения к серверу.

**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 клиент-серверных приложений, оформления графических интерфейсов с помощью JSX-компонентов и элементов, взаимодействия с базой данных sqlite, ведения Git-репозитория на сервисе github, размещения рабочих проектах на сервисах Heroku и Vercel. Также были получены базовые знания библиотеки построения графических интерфейсов React и расширения данной библиотеки – Next.js. Изучен фреймворк для написания серверной логики – Nest.js в связке с СУБД sqlite.

**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 c. — 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 c.— 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 c. — 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*

1. 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> .
2. Моделирование систем [Текст]: учебник для вузов / С.И. Дворецкий, Ю.Л. Муромцев, В.А. Погонин, А.Г. Схиртладзе. – М.: Академия, 2009. – 320 с.
3. Порсев, Е. Г. Организация и планирование экспериментов [Электронный ресурс]: учебное пособие / Е. Г. Порсев. — Новосибирск : Новосибирский государственный технический университет, 2010. — 155 c. — 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\_name':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="Cоздать 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>