## Введение

### Актуальность

Глубоко проработанный интерфейс является одним из главных составляющих успешного приложения. В первую очередь потому, что именно с ним происходит взаимодействие конечного пользователя продукта, а значит, он оказывает сильное влияние на оценку продукта.

Чтобы совмещать в интерфейсе широкий функционал и простоту для пользователя, нужно, по крайней мере, знать базовые подходы к его проектированию. Поэтому в данной практической работе мы исследуем принципы создания макетов, инструменты для разработки эффективных и привлекательных интерфейсов.

### Цель

Цель практической работы — в ходе командной работы приобрести навыки и знания в области создания интерфейсов и макетов, а также изучить инструменты, которые для этого принято использовать.

### Задачи

1. Выполнить обзор литературы, чтобы собрать информацию об интерфейсах, макетах и инструментах для их реализации.
2. Рассмотреть 3-4 инструмента и сравнить их по критериям: функциональность, удобство использования, качество результатов, совместимость и интеграция, цена.
3. Используя любой инструмент, создать макет интерфейса приложения «ToDoList», описывая обоснование этого интерфейса, основанное на принципах и подходах при создании интерфейсов.
4. Изучить тестирование макетов среди пользователей и анализ полученной обратной связи.
5. Выбрать предполагаемый стек технологий, который можно использовать для создания приложения «ToDoList».
6. Обобщить результаты проделанной работы и сделать выводы по теме исследования.

## Методология

Методология проекта представляет собой каскадную модель, включающую в себя следующие этапы:

* Аналитика;
* Проектирование;
* Разработка;
* Тестирование;
* Релиз.

Была выбрана такая методология, т.к. каскадная методология позволяет четко сформулировать все требования и спланировать все детали заранее, таким образом мы сразу понимаем, какой проект получится в итоге. Также такая методология делает упор на исчерпывающую документацию. В таком небольшом проекте нет необходимости в использовании Agile методологии.1

## Аналитика

### Требования к продукту2

#### **Что нужно сделать**:

* Сервер и база данных запускаются на удаленном сервере;
* Создать базу данных на основе определенной блок-схемы;
* Сделать макет интерфейса;
* Разработать фронтэнд часть приложения.

#### Для чего:

* Нужно создать веб-приложение todo list.

#### Дедлайн:

* Деплой веб сервиса намечен на конец октября этого года.

#### Фронтэнд часть должна выполнять следующие задачи:

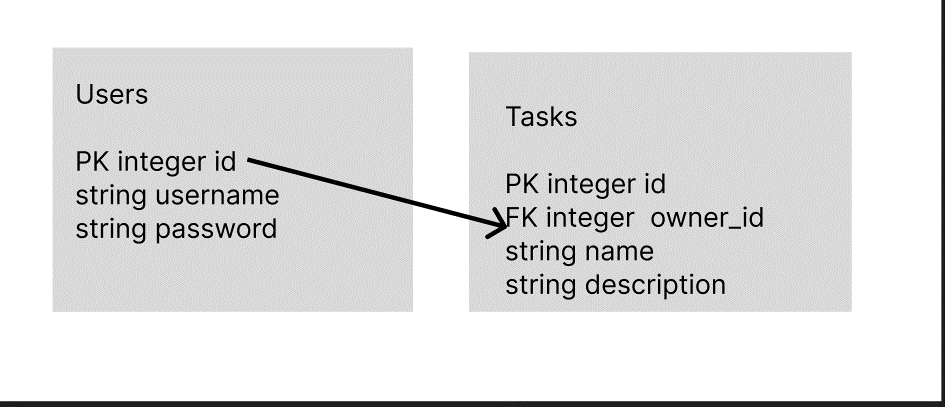
* Просмотр своих задач
* Добавление новых задач
* Удаление существующих задач
* Редактирование задач

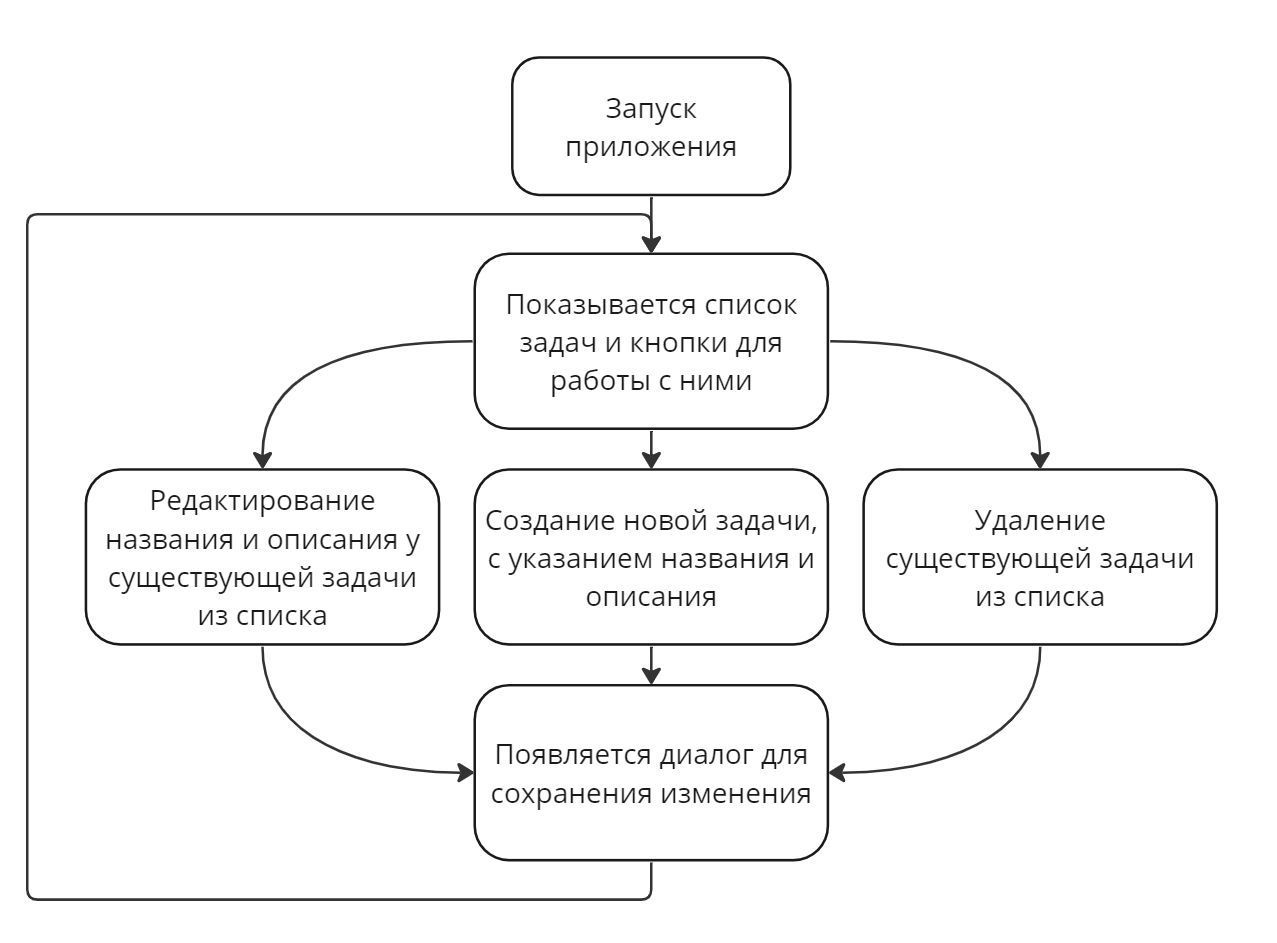
#### Бэкэнд часть берет на себя следующие обязанности:

* Хранение информации в базе данных
* Предоставление информации для фронтэнд части
* Редактирование информации в базе

## Проектирование

Схема базы данных



Алгоритм пользования приложением

### Разработка

Создать макет интерфейса

### Тестирование

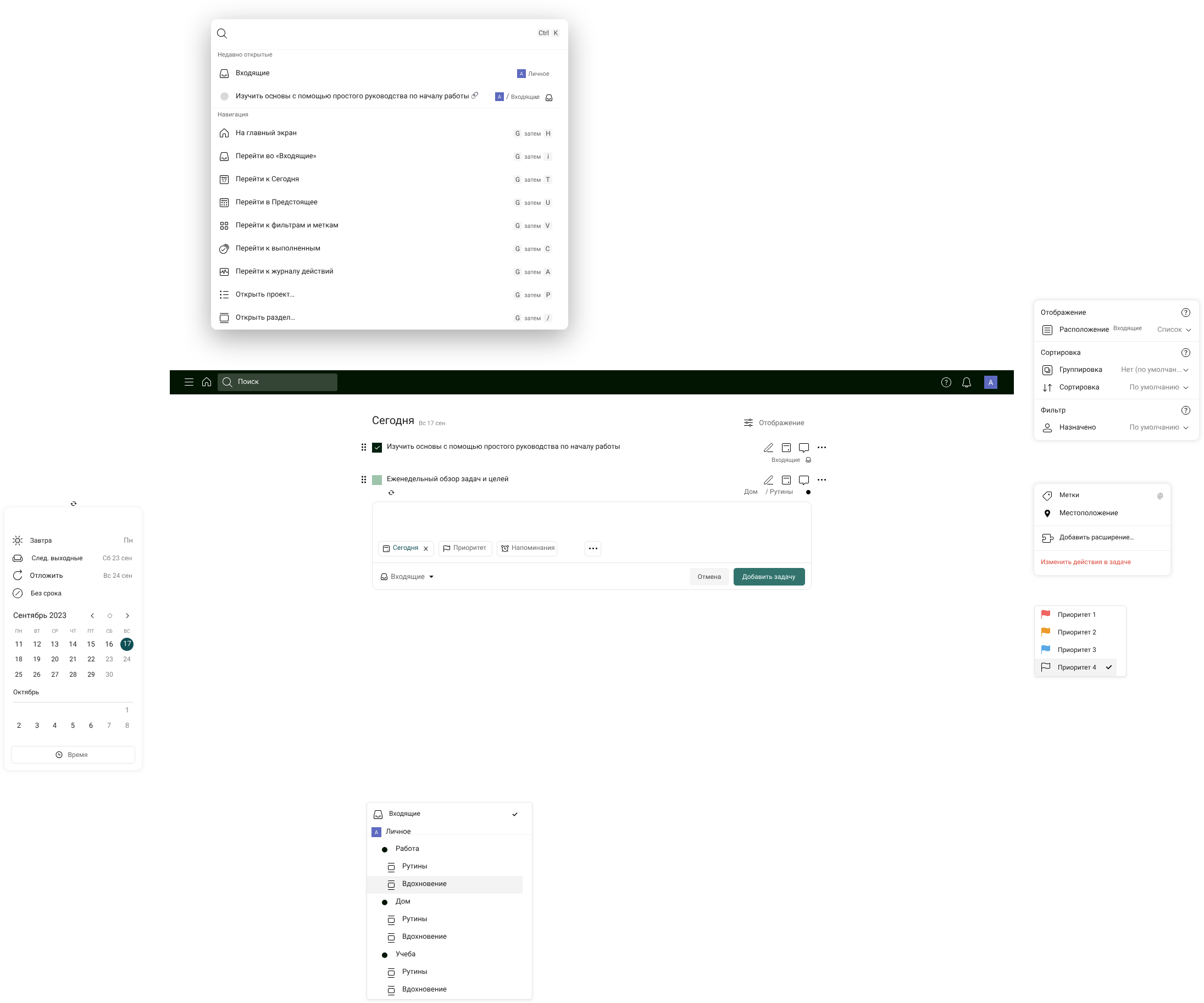
Продумать несколько сценариев, которые могут произойти при пользовании приложением, обозначить ожидаемый результат, провести ручное тестирование и сравнить полученный результат с ожидаемым.

Составить на основе тестирования таблицу для удобного анализа. Итоговая таблица должна занимать около 2 страниц.

### Релиз

Бэкэнд часть находится на удаленном сервере, база данных на том же сервере, фронтэнд часть представляет собой веб-сайт.

## Создание макетов



## Сравнительный анализ интерфейсов

| Характеристика | React | Angular | Vue.js | Svelte | Node.js | Flask |
| --- | --- | --- | --- | --- | --- | --- |
| Фреймворк/библиотека | React | Angular | Vue.js | Svelte | Express.js | Flask |
| Язык программирования | JavaScript/TS | TypeScript | JavaScript/TS | JavaScript/TS | JavaScript/TS | Python |
| Управление состоянием | Redux, Mobx, Context API | RxJS, NgRx | Vuex | Svelte Stores | N/A | N/A |
| Маршрутизация | React Router | Angular Router | Vue Router | SvelteKit Router | Express Router | Flask-Routing |
| HTTP-запросы | Axios, Fetch API | HttpClient | Axios, Vue Resource | Fetch API, библиотеки | Axios, Fetch API | Flask HTTP |
| Стилизация | CSS, SCSS, CSS-in-JS | CSS, SCSS | CSS, SCSS | CSS, SCSS | CSS, SCSS | CSS |
| Сборка и управление зависимостями | Webpack, Babel | Angular CLI | Vue CLI, Webpack | SvelteKit, Snowpack | N/A | N/A |
| Тестирование | Jest, Enzyme, React Testing Library | Jasmine, Karma, Protractor | Jest, Vue Test Utils | Svelte Testing Library | Jest, Mocha, Chai | pytest, unittest |
| Инструменты разработчика | React DevTools | Angular DevTools | Vue DevTools | Svelte DevTools | N/A | Flask Debug |
| Интеграция с бэкендом | REST API, GraphQL | REST API, GraphQL | REST API, GraphQL | REST API, GraphQL | REST API, GraphQL | REST API, GraphQL |
| Бэкенд-фреймворк | N/A | N/A | N/A | N/A | Express.js | Flask |
| Шаблонизатор | N/A | N/A | N/A | N/A | N/A | Jinja2 |

**Аргументы за использование каждого стека для to-do list:**

1. **React:**
   * Простой в использовании и быстрый для создания интерфейса.
   * Широкое сообщество и множество сторонних библиотек.
   * Хорошая поддержка состояния компонентов.
   * React Router для управления маршрутами.
2. **Angular:**
   * Интегрированный фреймворк с множеством функциональности.
   * Мощный инструментарий для управления состоянием.
   * Angular Router для маршрутизации.
3. **Vue.js:**
   * Простота использования и понимания.
   * Официальный набор инструментов для разработчиков.
   * Виртуальный DOM для оптимизации производительности.
4. **Svelte:**
   * Очень быстрый и легкий фреймворк.
   * Нет виртуального DOM, что способствует высокой производительности.
   * Простой синтаксис и интуитивно понятное обновление состояния.
5. **Node.js (Express.js):**
   * Подходит для создания серверной части приложения.
   * Широко используется для создания API.
   * Множество пакетов и библиотек для обработки HTTP-запросов и роутинга.
6. **Flask:**
   * Простой и легковесный Python-фреймворк для веб-разработки.
   * Хорошо подходит для создания API и маленьких веб-приложений.
   * Интеграция с Jinja2 для шаблонизации.

Почему выбрали React3:

1. **Производительность**:
   * React использует виртуальную DOM, что позволяет минимизировать количество фактических манипуляций с реальным DOM. Это приводит к лучшей производительности при обновлении интерфейса.
   * В сравнении с некоторыми альтернативами, такими как jQuery, React обычно имеет более высокую производительность благодаря этой оптимизации.
2. **Сообщество и экосистема**:
   * React имеет огромное активное сообщество разработчиков, библиотеки и решения для практически любой задачи легко доступны.
   * Сравнивая с Vue.js, React имеет более широкий набор сторонних библиотек и интеграций.
3. **Инструменты для разработки**:
   * React поддерживается множеством инструментов для отладки, таких как React DevTools и Redux DevTools, что упрощает процесс разработки и отладки.
   * В сравнении с Vanilla JavaScript, разработка на React часто более эффективна благодаря использованию компонентов и инструментам.
4. **Модульность и переиспользование кода**:
   * React поощряет модульность и переиспользование кода благодаря компонентной архитектуре. Это делает его отличным выбором для приложений с множеством интерфейсных элементов, таких как To-Do List.
   * В сравнении с Vanilla JavaScript, где нет строгой структуры, React позволяет легко масштабировать и поддерживать код.
5. **Сравнение с альтернативами**:

**Angular**: React обычно считается более простым для изучения и быстрым для разработки по сравнению с Angular, который имеет более высокий порог входа и более сложную архитектуру.

* + **Vue.js**: React и Vue.js более схожи в популярности, но React имеет более широкую экосистему и поддержку от крупных компаний, таких как Facebook.
  + **jQuery**: React предоставляет более современный и эффективный подход к разработке интерфейсов по сравнению с jQuery, который работает непосредственно с DOM и менее подходит для больших и сложных приложений.

1. **Гибкость**:
   * React не навязывает какую-либо строгую архитектуру, что делает его гибким для разных типов проектов.
   * Сравнивая с Ember.js, который имеет более строгую структуру, React может легче адаптироваться к различным потребностям.

В итоге, выбор React для создания To-Do List обоснован его высокой производительностью, богатой экосистемой, модульностью и поддержкой со стороны сообщества. Также он предоставляет удобные инструменты для разработки и поддержки приложения, что делает его хорошим выбором для такого проекта.

## **Тестирование4**

**Тестирование программного обеспечения (Software Testing) —** проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом.

**В чем цель тестирования?**

**Цель тестирования —** проверка соответствия ПО предъявляемым требованиям, обеспечение уверенности в качестве ПО, поиск очевидных ошибок в программном обеспечении, которые должны быть выявлены до того, как их обнаружат пользователи программы.

**Для чего проводится тестирование ПО?**

* Для проверки соответствия требованиям.
* Для обнаружения проблем на более ранних этапах разработки и предотвращение повышения стоимости продукта.
* Обнаружение вариантов использования, которые не были предусмотрены при разработке. А также взгляд на продукт со стороны пользователя.
* Повышение лояльности к компании и продукту, т. к. любой обнаруженный дефект негативно влияет на доверие пользователей.

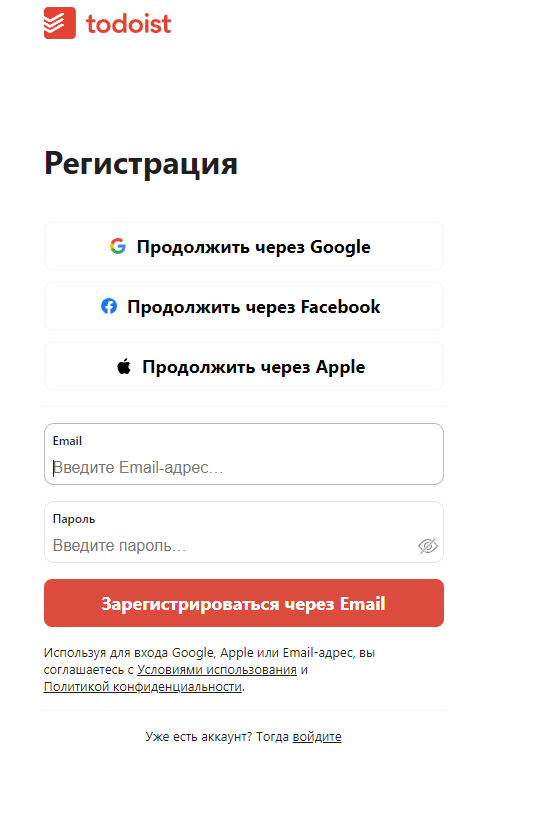
**Современные способы тестирования.5**

На данный момент существует несколько популярных способов тестирования пользовательских интерфейсов:

1. **Ручное тестирование** - это метод, при котором тестировщики вручную проверяют интерфейс на соответствие требованиям и оценивают его работоспособность. В процессе ручного тестирования тестировщик взаимодействует с интерфейсом, выполняет определенные действия и проверяет, что результаты соответствуют ожиданиям. Ручное тестирование позволяет проверить различные сценарии использования, а также оценить визуальный дизайн и удобность интерфейса.
2. **Автоматизированное тестирование** - это использование специальных инструментов и написание скриптов, которые выполняют определенные тестовые сценарии автоматически. Тестовые скрипты могут симулировать действия пользователя, такие как нажатие кнопок, заполнение форм, навигацию по страницам и т.д. С помощью автоматизированного тестирования можно ускорить процесс тестирования, снизить вероятность человеческой ошибки и повысить надежность результатов.
3. **Тестирование с использованием анализа пользовательских метрик и данных**:
4. В этом виде тестирования данные о пользовательском поведении собираются и анализируются с помощью машинного обучения и статистических методов. Например, можно анализировать данные о времени, проведенном пользователями на страницах, пути продвижения пользователя по сайту, клики на определенные элементы интерфейса и другие метрики. Это может помочь выявить проблемы в интерфейсе, такие как сложности в навигации или непонятные элементы, и предложить улучшения на основе реальных данных.
5. **Тестирование интерфейса пользователя (UI тестирование)—**

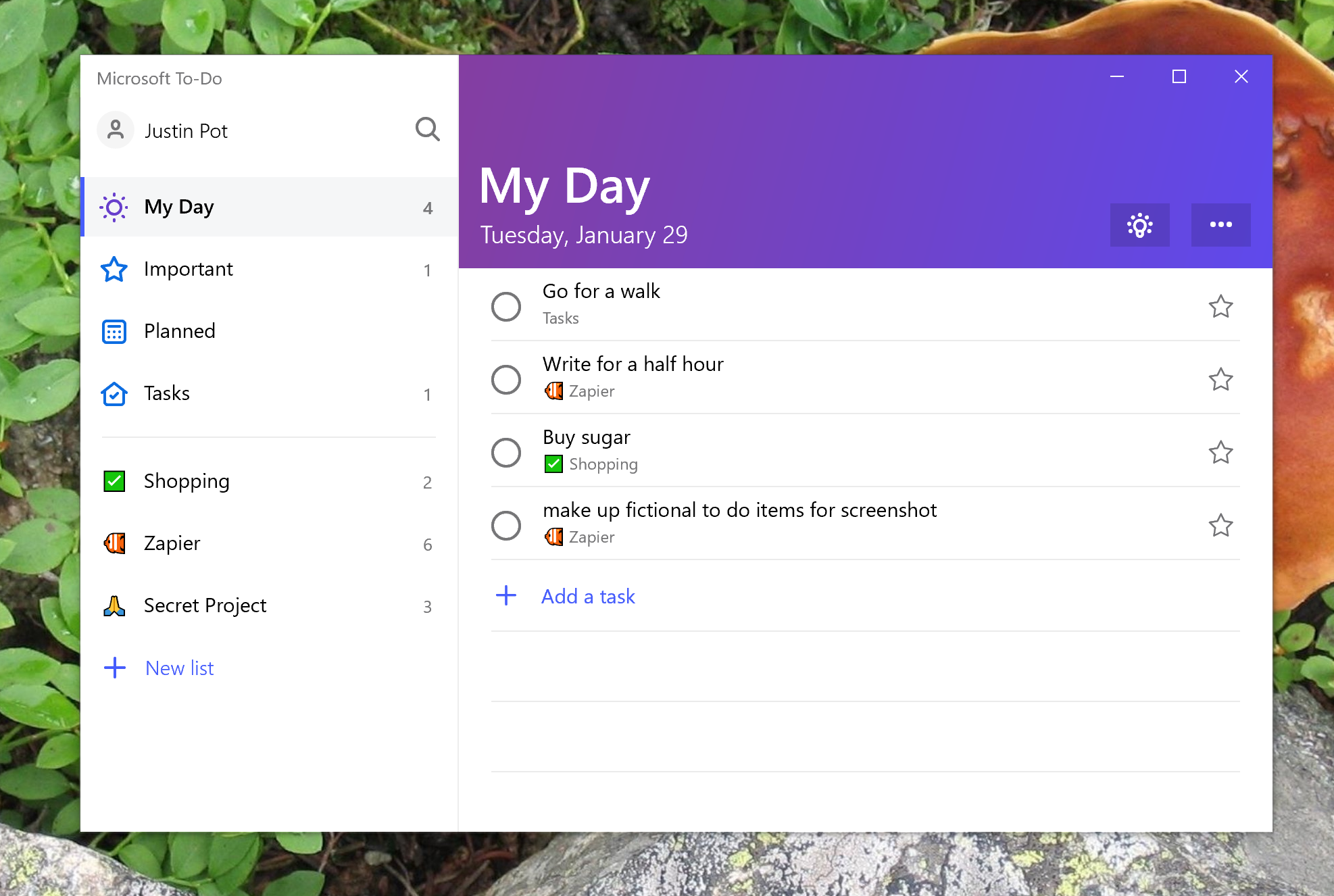
это процесс проверки графического интерфейса программного приложения с целью обнаружения ошибок и неполадок. Оно осуществляется путем автоматизации действий пользователя на интерфейсе приложения и последующего анализа результатов.  
UI-тестирование позволяет проверить, что интерфейс приложения соответствует требованиям пользователей и спецификациям. Оно включает в себя проверку внешнего вида, расположения и взаимодействия элементов интерфейса, корректность отображения данных, работу переходов между экранами и другие аспекты, которые влияют на пользовательский опыт.

**Пример UI-тестирования на примере приложения ToDoList.**

**Рисунок 1: Пример стартового экрана приложения.**

* Проверка работоспособности кнопок «Продолжить через Google», «S Продолжить через Facebook», « Продолжить через Apple».
* Проверка ввода данных в всплывающие окна.
* Проверка ввода некорректных данных в всплывающие окна.
* Проверка, что приложение сохраняет данные пользователя после выхода из приложения.
* Проверка работоспособности гиперссылок «Условия использования» и «Политика конфиденциальности».

**Рисунок 2: Пример главного экрана приложения.**



* Проверка работоспособности кнопки «Добавить новый список» или другой аналогичной кнопки для выполнения этой команды.
* Проверка наличия поля для ввода новых задач.
* Ввод новой задачи в поле ввода.
* Проверка работоспособности кнопки «Добавить» или другой аналогичной кнопки для добавления задачи.
* Проверка появления задачи в списке задач.
* Проверка возможности пометить задачу как выполненную.
* Проверка, что выполненная задача отображается соответственно или отображается отдельно.
* Проверка возможности удалить задачу.
* Проверка, что задача успешно удаляется и исчезает из списка задач.
* Проверка, что приложение обрабатывает ситуации с некорректными вводными данными, например с пустым полем.
* Проверка, что введенная задача сохраняется после перезагрузки приложения.
* Проверка редактирования задачи.
* Проверка сохранения и отображения задачи в списке.
* Проверка возможности фильтрации задач по разным категориям или приоритетам.
* Проверка сортировки задач по датам и другим параметрам.
* Проверка, что приложение корректно отображается на разных устройствах и разрешениях экрана.
* Проверка возможности импорта\экспорта списка задач в различных форматах(например, CSV или JSON).
* Проверка, что приложение корректно обрабатывает ошибки и выводит соответствующие сообщения об ошибках.
* Проверка работоспособности разных вкладок с задачами.
* Проверка возможности сменить аккаунт.

После тестирования и вывода продукта на рынок необходима обратная связь от пользователей.

**Зачем нужна обратная связь?6**

Сбор обратной связи, анализ отзывов и изменения в услуге на основе полученной информации помогают удерживать старых и привлекать новых клиентов, в конечном счете отражаясь на прибыльности бизнеса.

Существует несколько способов получения и интерпретации обратной связи.

Они могут быть разными в зависимости от контекста и цели обратной связи.

1. **Анкеты и опросы**: Этот метод включает создание структурированных вопросов и их распространение среди целевой аудитории. Анкеты могут быть распространены как в печатном формате, так и в электронном виде через электронную почту, социальные сети или специализированные платформы для проведения опросов. Анкеты могут быть как открытыми (с возможностью свободного ответа), так и закрытыми (с предложенными вариантами ответов). После получения ответов, результаты анкеты анализируются для выявления общих тенденций и взглядов аудитории.
2. **Фокус-группы**:Этот метод включает организацию групповых дискуссий с небольшой группой представителей целевой аудитории. Участники фокус-группы могут обсуждать определенную тему, задавать вопросы друг другу и делиться своими мнениями и отзывами. Фокус-группы могут проводиться лично, по телефону или онлайн через видеоконференции. Результаты дискуссий могут быть записаны и проанализированы для получения ценных выводов.
3. **Открытые форумы:** Организация открытых форумов или платформ, где люди могут свободно высказывать свое мнение и делиться отзывами и идеями. Это может быть веб-сайт, специальный онлайн-форум, чат-бот или социальная сеть. Открытые форумы позволяют людям общаться между собой, задавать вопросы, отвечать на них и обмениваться идеями. Результаты открытых форумов могут быть анализированы для выявления общих мнений и предпочтений.
4. **Обратная связь через социальные сети:** Отслеживание и анализ отзывов и комментариев, оставленных людьми на платформах социальных сетей, таких как Вконтакте, Telegram, Twitter и др. Мнения и отзывы, оставленные публично, могут быть анализированы для определения общего мнения пользователей о продукте, бренде или услуге. Это также может помочь в выявлении проблем и сильных сторон продукта или услуги.
5. **Клиентские обзоры и оценки:** Изучение обзоров, оставленных клиентами на различных платформах, таких как Google, Yelp, Tripadvisor и другие. Клиенты имеют возможность оставить отзывы и оценки после использования продукта или услуги. Это дает возможность получить представление о том, что клиенты думают о вашей компании или продукте.

Интерпретация обратной связи требует анализа полученных данных и выделения ключевых выводов. Важно обратить внимание не только на конкретные мнения, но и на общие тенденции и паттерны. Кроме того, автоматизация обработки обратной связи с помощью машинного обучения и анализа данных может существенно упростить этот процесс.

## Заключение

В ходе практической работы студенты приобрели навыки и знания в области создания интерфейсов и макетов, а также изучили инструменты, которые для этого принято использовать. Также были рассмотрено тестирование интерфейсов и возможные пользовательские сценарии.

Был выбран стек технологий для создания приложения ToDoList: React, Express.js и PostgreSQL.

Ссылка на github: https://github.com/DaniilStolyarov/Interface-overview

## Список использованных источников

1. Методология управления IT-продуктом // VC.RU URL: https://vc.ru/hr/307719-metodologiya-upravleniya-it-produktom (дата обращения: 17.09.2023).

2. Как правильно сформировать требования к ИТ-системе // VC.RU URL: https://vc.ru/u/224864-viktor-nikitin/643415-kak-pravilno-sformirovat-trebovaniya-k-it-sisteme (дата обращения: 17.09.2023).

3. Как правильно сформировать требования к ИТ-системе // Хабр URL: https://habr.com/ru/companies/auriga/articles/703836/ (дата обращения: 17.09.2023).

4. Фундаментальная теория тестирования // Хабр URL: https://habr.com/ru/articles/549054/ (дата обращения: 17.09.2023).

5. Виды и типы тестирования: подробный разбор // TESTENGINEER URL: https://testengineer.ru/vidy-tipy-testirovaniya/ (дата обращения: 17.09.2023).

6. Фидбэк: что это такое, какие виды существуют // Академия Продаж URL: https://academy-of-capital.ru/blog/fidbek/ (дата обращения: 17.09.2023).