**СОДЕРЖАНИЕ**

[Введение 1](#_Toc133631642)

[1. Системы управления проектами 2](#_Toc133631643)

[2. Анализ методологий разработки 3](#_Toc133631644)

[3. Анализ существующих решений 4](#_Toc133631645)

[4. Анализ платформ разработки 7](#_Toc133631646)

[Заключение 8](#_Toc133631647)

[Список литературы 9](#_Toc133631648)

Введение

В наше время бизнес разработки продуктов является одной из самых перспективных отраслей экономики. Благодаря постоянному развитию технологий и изменению потребительских предпочтений, появляются новые возможности для создания и продвижения инновационных продуктов на рынке.

Одним из ключевых факторов успеха в бизнесе разработки продуктов является умение быстро адаптироваться к изменяющимся условиям и требованиям рынка. Для этого необходимо постоянно улучшать процессы разработки, используя современные методологии и инструменты управления проектами.

Системы управления проектами играют важную роль в современном бизнесе, помогая организациям эффективно планировать, контролировать и реализовывать проекты различной сложности. Они помогают сокращать издержки и время на реализацию проектов [1], улучшать коммуникацию и сотрудничество между участниками проекта, управлять рисками и улучшать качество продуктов и услуг. Системы управления проектами также позволяют улучшить управление ресурсами, такими как время, бюджет и человеческие ресурсы.

Данные системы могут быть использованы в различных отраслях, таких как IT, строительство, производство, маркетинг, финансы и другие [2]. Они могут быть настроены для соответствия требованиям конкретной отрасли и размеру проекта, что делает их универсальными инструментами при разработке почти любых типов продуктов.

1. Системы управления проектами

Система управления проектами (Project Management System) – это комплекс программных и методологических средств, используемых для планирования, управления и контроля выполнения проектов [3]. Она включает в себя набор процессов, инструментов и техник, которые помогают организациям достигать своих целей, оптимизировать использование ресурсов и минимизировать риски.

Эффективное использование систем управления проектами позволяет достичь поставленных целей и задач в срок, снизить риски и улучшить качество продукта или услуги. Без подобных систем компании могут столкнуться с проблемами, такими как нехватка ресурсов, неопределенность целей и задач, превышение бюджета и неудовлетворительное качество работы.

Системы управления проектами очень актуальны в наше время во многих сферах, поэтому целью данной работы является их изучение.

Цель:

Изучение систем управления проектами

Задачи:

* Провести анализ методологий разработки систем управления проектами;
* Провести анализ существующих решений;
* Провести анализ платформ разработки систем управления проектами.

1. Анализ методологий разработки

Существует множество методологий управления проектами, каждая из которых представляет собой набор правил, процедур и методов, направленных на достижение целей проекта и управление его рисками. Рассмотрим несколько наиболее распространенных методологий [4]:

1. Agile – это методология разработки продуктов, которая основана на гибком и итеративном подходе к управлению проектами [5-6]. Она предполагает частые итерации и постоянное общение с заказчиком, чтобы быстро адаптироваться к изменениям требований и получать обратную связь. Agile используется в различных областях, таких как IT, маркетинг, производство и т. д.
2. Waterfall – это классическая последовательная методология, в которой каждая фаза проекта выполняется последовательно и завершается до перехода к следующей [7-8]. Она используется для проектов, где требования и цели четко определены и не меняются в процессе разработки. Преимуществом методологии Waterfall является ее предсказуемость и понятность, а недостатком – жесткость и невозможность быстро адаптироваться к изменениям.
3. Lean – это философия управления, которая ставит целью минимизацию потерь в процессах и повышение эффективности работы команды. Lean используется для управления процессами производства и предполагает постоянный анализ и оптимизацию процессов, чтобы устранить неэффективные шаги и увеличить производительность.
4. Scrum – это один из подходов Agile, который основан на коротких циклах разработки, называемых спринтами. Он используется для проектов, где цели и требования могут меняться в процессе разработки. Преимуществом методологии Scrum является ее гибкость и возможность быстрой адаптации. К недостаткам можно отнести необходимость строгой организации команды и выполнения спринтов в срок.
5. Kanban – это методология управления проектами, основанная на визуальном отображении рабочего процесса и управлении потоком работы. Она используется для улучшения эффективности проекта и сокращения времени, необходимого для выполнения задач [9-10]. Преимуществом методологии Kanban является ее простота и гибкость. Недостатком является отсутствие достаточного контроля над проектом и сложность визуализации проектов большой сложности.

Каждый из этих подходов имеет свои преимущества и недостатки, и может использоваться в различных областях и проектах [11]. Однако, в зависимости от проекта и задачи, может потребоваться комбинирование этих методологий для достижения наилучших результатов.

1. Анализ существующих решений

Существует множество систем управления проектами (СУП), которые помогают организовывать и контролировать выполнение проектов, снижать риски и повышать эффективность работы команды. Каждые из них имеют свои преимущества и недостатки, которые определяют, насколько они подходят для разработки продуктов в различных сферах.

Jira

Jira – это СУП, которая часто используется для разработки программного обеспечения (рис.1). Она позволяет организовывать и контролировать задачи, баги, выпуски, релизы и т. д. в рамках Agile-методологий разработки [12].

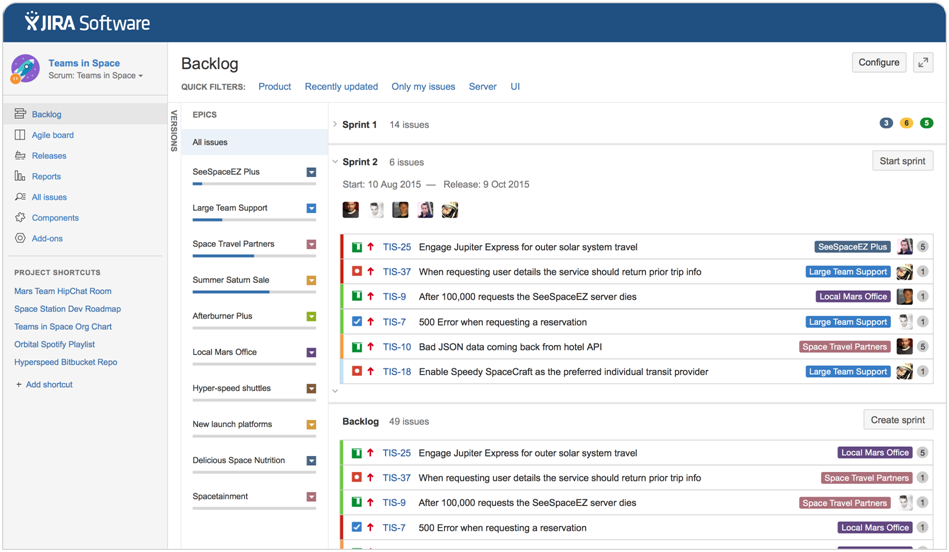


Рис.1 Интерфейс системы Jira

**Преимущества:**

1. Гибкие настройки, благодаря которым систему можно настроить под конкретные нужды команды;
2. Возможность отслеживать и визуализировать прогресс выполнения задач;
3. Возможность интеграции системы с другими инструментами разработки (Git, Confluenceи т. д.).

**Недостатки:**

1. Может быть сложной в освоениииз-за большого количества функций, что способно замедлить внедрение системы;
2. Проблемы с производительностью при большом количестве пользователей и задач;
3. Мало возможностей для автоматизации работы с системой.

Система лучше всего подходит для разработки программного обеспечения. Отлично справляется с большим количеством пользователей и задач. Также она имеет большое количество важных функций, способствующих комфортной работе всей команды, но из-за этого может быть трудна в освоении.

Trello

Trello – это простая и интуитивно понятная СУП(рис.2), которая широко применяется для управления проектами небольших команд или стартапов [13]. Она позволяет организовывать и контролировать задачи и процессы в удобном виде, используя карточки, списки и доски.

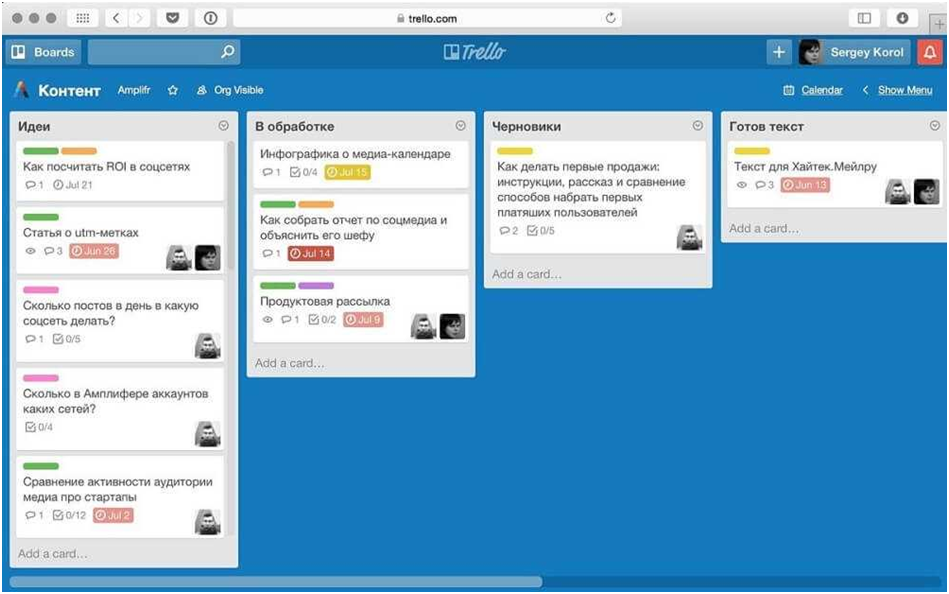


Рис.2 Интерфейс системы Trello

**Преимущества:**

1. Легкая и быстрая настройка, простота в использовании;
2. Гибкость, возможность подстроить систему под определенные потребности;
3. Возможность интеграции с другими инструментами разработки (GoogleDrive, Slack, Zapier).

**Недостатки:**

1. Ограниченный функционал системы, нет большого количества функций для комфортной работы;
2. Низкий уровень контроля и наблюдения за выполнением задач;
3. Трудности при работе с большим количеством проектов.

Система является очень простой в использовании, не обладая при этом большим количеством функций. Поэтому она наиболее подходит для совместной работы небольших команд, а также для личного пользования.

Asana

Asana – это СУП, которая позволяет организовывать и контролировать задачи и процессы на основе принципов Lean-методологии(рис.4). Она имеет широкий набор функций, таких как управление задачами, проектами, командами и т. д. [14].

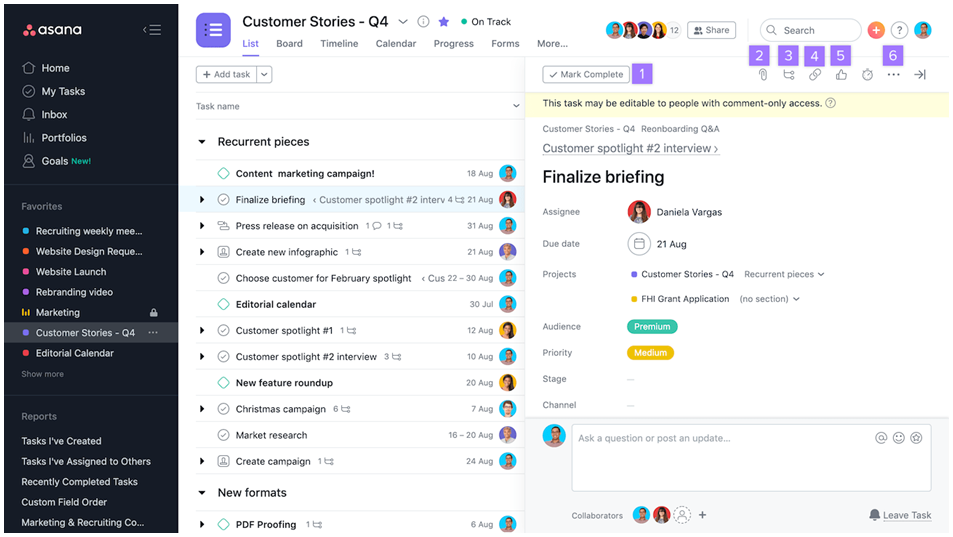


Рис.3 Интерфейс системы Asana

**Преимущества:**

1. Интуитивно понятный интерфейс;
2. Есть возможность добавления соисполнителей и наблюдателей в задачи;
3. Множество дополнительных удобных функций (календарь, напоминания, совместное использование документов и другие).

**Недостатки:**

1. Нет возможности добавление подзадач, что может быть неудобно при работе с крупными проектами;
2. Нет иерархии сотрудников;
3. Неудобства в работе с большим количеством задач.

Система оптимальна для команд и отделов среднего размера, подходит для работы в различных сферах. Имеет понятный и удобный интерфейс. При этом система не обладает функциями, предназначенными для разработки продуктов в отдельных сферах, например для разработки программного обеспечения.

1. Анализ платформ разработки

Существуют различные типы программного обеспечения, которые выполняются на разных устройствах и используются для разных целей.

Десктопные приложения, также известные как приложения для настольных компьютеров, запускаются и работают на компьютере пользователя, без необходимости доступа к интернету. Такие приложения могут иметь более широкие функциональные возможности, чем мобильные и веб-приложения, и могут использоваться для более сложных задач, таких как обработка данных, проектирование и разработка, или для использования приложений, связанных с сенсорными устройствами и оборудованием, подключенными к компьютеру.

Мобильные приложения разработаны для использования на мобильных устройствах, таких как смартфоны и планшеты. Они часто имеют более узкую функциональность, чем десктопные приложения, и направлены на решение более специфических задач, таких как социальные медиа, банковские операции или организация задач и управление проектами. Важным фактором для мобильных приложений является оптимизация интерфейса и удобство использования на маленьких экранах.

Веб-приложения работают в браузере и требуют подключения к интернету. Они могут быть доступны через различные устройства и операционные системы, что делает их удобными для использования на любом устройстве с доступом в Интернет. Веб-приложения могут быть разработаны для разных целей, таких как электронная коммерция, облачное хранение данных или онлайн-сервисы, такие как электронная почта или календарь.

Каждый тип приложения имеет свои преимущества и недостатки, и выбор зависит от конкретных потребностей пользователя и задач, которые необходимо решить.

Системы управления проектами могут использоваться на различных устройствах, включая персональные компьютеры, ноутбуки, планшеты и смартфоны.

Для более сложных задач, таких как разработка проектного плана или анализ рисков, удобнее использовать более крупные устройства, такие как ноутбуки и настольные компьютеры, для обеспечения максимальной производительности и удобства работы с большим объемом информации.

Однако для управления проектом в реальном времени, мониторинга задач и обновления информации на ходу, мобильные устройства, такие как планшеты и смартфоны, могут быть более удобными. Также, мобильные устройства могут использоваться для быстрого доступа к системе управления проектами, когда нет возможности работать на стационарном компьютере или ноутбуке.

Благодаря своей универсальности веб-приложения наиболее подходят для разработки систем управления проектами. Такую систему пользователь сможет использовать на любом типе устройства, в любое время, и в любом удобном для него месте, где есть доступ в интернет.

1. Анализ языков программирования Front-End

Клиентская часть веб-приложений отвечает за взаимодействие с пользователем и отображение данных в браузере.

Основой для разработки веб-приложений является язык разметки HTML (HyperText Markup Language), необходимый для создания структуры и содержимого веб-страниц, с помощью различных элементов и тегов, таких как заголовки, параграфы, списки, изображения и другие. Для оформления и стилизации веб-страниц используется CSS (Cascading Style Sheets), он позволяет создавать красивые и привлекательные пользовательские интерфейсы, а также обеспечивает адаптивность и отзывчивость веб-приложений. Без данных инструментов разработка веб-приложения невозможна, при этом они используются только вместе с одним из языков программирования.

Для разработки веб-приложений существуют всего два языка программирования:

1. JavaScript — мультипарадигменный язык программирования для клиентской части веб-приложений, позволяющий создавать интерактивные элементы, управлять динамическим содержимым веб-страницы, выполнять асинхронные запросы к серверу и многое другое. Он поддерживается всеми современными браузерами. JavaScript также используется во многих фреймворках и библиотеках.
2. TypeScript ‑ язык программирования, основанный на JavaScript, но с добавлением статической типизации. Он предоставляет разработчикам инструменты для разработки крупномасштабных приложений, облегчая обнаружение ошибок на этапе разработки и повышая поддержку кода. TypeScript компилируется в JavaScript и может использоваться вместе с различными фреймворками.

Несмотря на то, что TypeScript основан на JavaScript, они имеют некоторые отличия, такие как:

* Синтаксис и типизация. JavaScript является динамически типизированным языком программирования, что означает, что типы переменных определяются автоматически во время выполнения программы. Он не требует явного указания типов данных. TypeScript, с другой стороны, является статически типизированным языком, что означает, что типы переменных определяются на этапе компиляции. Он добавляет статическую типизацию к JavaScript, позволяя разработчикам указывать типы переменных, параметров функций и возвращаемых значений.
* Компиляция. JavaScript выполняется непосредственно в браузере или на сервере без предварительной компиляции. Он интерпретируется и выполняется по мере загрузки веб-страницы. TypeScript, как язык со статической типизацией, требует предварительной компиляции. Исходный код TypeScript компилируется в JavaScript перед его выполнением. Это означает, что перед использованием кода TypeScript необходимо его скомпилировать в JavaScript.
* Расширенные возможности. TypeScript предлагает дополнительные функции, которых нет в JavaScript. Например, TypeScript поддерживает статическую типизацию, что может помочь обнаруживать ошибки на этапе компиляции и повышает надежность кода. TypeScript также включает возможности использования классов, интерфейсов, перечислений и других объектно-ориентированных конструкций, которые упрощают разработку и поддержку сложных проектов.
* Совместимость. Поскольку TypeScript является надмножеством JavaScript, весь действительный код JavaScript является допустимым кодом TypeScript. Это означает, что вы можете постепенно внедрять TypeScript в существующий JavaScript-код и использовать их вместе.
* Экосистема и сообщество. JavaScript имеет обширную экосистему с множеством библиотек, фреймворков и инструментов, доступных для разработчиков. Он является одним из самых популярных языков программирования веб-разработки. TypeScript также имеет широкую поддержку и активное сообщество разработчиков. Он интегрируется с различными инструментами разработки, такими как Visual Studio Code, и широко используется в крупных проектах.

Благодаря их отличиям, можно выделить плюсы и минусы для обоих языков программирования.

|  |  |  |
| --- | --- | --- |
| **Язык программирования** | **JavaScript** | **TypeScript** |
| **Плюсы** | Универсальность; Большая экосистема; Простота изучения. | Статическая типизация; Улучшенная разработка; Совместимость с JavaScript. |
| **Минусы** | Динамическая типизация; Отсутствие строгих правил; Проблемы с безопасностью. | Необходимость компиляции; Ограничения экосистемы; Сложность изучения. |

TypeScript предоставляет дополнительные функции и инструменты, которые могут улучшить процесс разработки и поддержки больших проектов. Однако выбор между JavaScript и TypeScript зависит от конкретных требований проекта, опыта разработчика и предпочтений команды разработки.

Вывод

1. Анализ фреймворков Front-End

JavaScript-фреймворк ‑ это набор предопределенных структур, библиотек и инструментов, созданных на основе языка JavaScript, с целью упростить разработку веб-приложений. Фреймворк предоставляет разработчикам готовые решения и стандартизированные подходы к разработке, чтобы ускорить процесс и облегчить задачи, связанные с созданием сложных веб-приложений.

Фреймворки обычно предлагают ряд функциональных возможностей и инструментов, таких как:

* Архитектура приложения. Фреймворки обычно предлагают определенную архитектуру приложения, которая помогает организовать код и структуру приложения. Например, многие фреймворки используют концепцию компонентов или модулей для организации кода.
* Маршрутизация. Фреймворки часто предлагают механизм маршрутизации, который облегчает управление маршрутами и навигацией в приложении. Это позволяет создавать SPA (одностраничные приложения), где содержимое обновляется динамически без перезагрузки страницы.
* Управление состоянием. Многие фреймворки предлагают инструменты для управления состоянием приложения. Они обеспечивают механизмы для хранения, изменения и обновления данных приложения на разных уровнях.
* Визуальные компоненты. Фреймворки предоставляют набор предопределенных компонентов и UI-элементов, которые можно использовать для создания пользовательского интерфейса. Это помогает ускорить разработку и обеспечить единообразный внешний вид и поведение интерфейса.
* Инструменты для тестирования и отладки. Многие фреймворки предлагают инструменты для автоматизации тестирования приложения и обнаружения и исправления ошибок.

JavaScript имеет множество фреймворков, которые облегчают разработку веб-приложений. Наиболее популярными из них являются React, Angular и Vue. При этом данные фреймворки сильно различаются по многим аспектам, а также имеют свои плюсы и минусы.

React

React (или React.js) ‑ это библиотека JavaScript, разработанная компанией Facebook, которая используется для создания пользовательских интерфейсов (UI) веб-приложений. Она позволяет разработчикам строить компоненты, которые представляют собой небольшие, изолированные части пользовательского интерфейса, и затем объединять их в сложные приложения.

React использует концепцию виртуального DOM (Document Object Model), что означает, что он создает в памяти представление пользовательского интерфейса и эффективно обновляет только измененные части, минимизируя количество операций на реальном DOM. Это делает React очень эффективным и быстрым в работе.

React также предоставляет механизм управления состоянием приложения, который позволяет легко отслеживать и обновлять данные во время выполнения приложения. Он позволяет создавать динамические интерфейсы, которые реагируют на пользовательские действия или изменения данных.

React можно использовать для разработки одностраничных приложений (SPA), мобильных приложений, десктопных приложений и других типов веб-приложений. Он широко используется в индустрии разработки веб-приложений и имеет большую и активную сообщество разработчиков.

Angular

Angular (или Angular.js) ‑ это фреймворк для разработки веб-приложений, разработанный компанией Google. Он предоставляет инструменты и функциональность для создания эффективных и масштабируемых одностраничных приложений (SPA).

Angular использует шаблонный подход для создания пользовательского интерфейса, основанный на расширении HTML с использованием директив и связывания данных (data binding). Он позволяет разработчикам создавать компоненты, которые представляют отдельные части пользовательского интерфейса, и объединять их для построения сложных приложений.

Основные особенности Angular:

* Компонентный подход. Приложение строится из набора компонентов, которые взаимодействуют друг с другом. Каждый компонент содержит свой HTML-шаблон, логику и стили.
* Двустороннее связывание данных. Angular предоставляет механизм двусторонней связи данных, который автоматически обновляет пользовательский интерфейс при изменении данных и наоборот.
* Внедрение зависимостей. Angular имеет встроенную систему внедрения зависимостей (Dependency Injection), которая упрощает управление зависимостями и создание тестируемого кода.
* Роутинг. Angular предоставляет возможность маршрутизации и навигации между различными компонентами и представлениями внутри приложения.
* Расширяемость. Angular обладает модульной архитектурой, которая позволяет добавлять сторонние модули и библиотеки для расширения функциональности приложения.

Angular написан на языке TypeScript, который предоставляет статическую типизацию и другие возможности, упрощающие разработку и обслуживание приложений.

Vue

Vue.js (или просто Vue) ‑ это прогрессивный JavaScript-фреймворк для разработки пользовательских интерфейсов. Он является открытым проектом и разрабатывается активным сообществом разработчиков. Vue позволяет создавать интерактивные веб-приложения, аналогично React и Angular.

Основные особенности Vue:

* Простота изучения. Vue обладает простым и интуитивным синтаксисом, что делает его легко изучаемым для новичков и быстро применимым в проектах.
* Компонентная архитектура. Vue основан на компонентах, которые могут быть созданы и переиспользованы для построения пользовательского интерфейса. Компоненты Vue включают шаблоны HTML, стили CSS и логику JavaScript.
* Реактивность. Vue обеспечивает реактивное поведение данных, что означает, что изменения данных автоматически отражаются в пользовательском интерфейсе без явного обновления. Это делает код более понятным и удобным для разработки.
* Директивы. Vue предоставляет множество встроенных директив, которые позволяют легко управлять поведением компонентов. Директивы позволяют добавлять интерактивность и реагировать на пользовательские события.
* Виртуальный DOM. Аналогично React, Vue использует виртуальный DOM для эффективного обновления только измененных частей пользовательского интерфейса.
* Расширяемость. Vue предоставляет возможность создавать собственные плагины и расширения для добавления дополнительных функций и интеграции с другими библиотеками.

Vue можно использовать в различных проектах ‑ от небольших интерактивных компонентов до больших масштабируемых приложений. Он также имеет официальные пакеты и инструменты для разработки мобильных приложений (Vue Native) и десктопных приложений (Vue.js + Electron).

Вывод

1. Анализ CSS-фреймворков Front-End

CSS-фреймворк (Cascading Style Sheets Framework) ‑ это набор предварительно написанных стилей CSS, который предоставляет готовые компоненты и соглашения о стилях для упрощения разработки веб-интерфейсов. Он облегчает создание современного и отзывчивого дизайна, а также повышает эффективность работы разработчиков, предоставляя готовые решения для общих задач.

Данные фреймворки обычно содержат набор классов CSS, которые можно применять к HTML-элементам для быстрой настройки их внешнего вида и расположения. Эти классы предопределены и стилизованы заранее, позволяя разработчикам создавать согласованный и профессиональный дизайн, не тратя много времени на написание стилей с нуля.

Также такие фреймворки могут быть полезными инструментами для быстрой разработки и обеспечения консистентности дизайна, особенно если у вас ограниченное время или опыт веб-разработки. Однако, при использовании фреймворков, важно понимать их особенности и правильно применять их классы и компоненты в соответствии с требованиями вашего проекта.

Существует большое количество различных CSS-фреймворков. Каждый предлагает свои уникальные возможности, набор компонентов и стилей, а также подходы к организации кода и адаптивности. Наиболее популярными из них являются:

1. Bootstrap ‑ это один из самых популярных и широко используемых CSS-фреймворков. Он предоставляет множество готовых компонентов, таких как кнопки, формы, навигационные меню, карусели и многое другое. Bootstrap также имеет мощную сеточную систему, которая помогает создавать адаптивные веб-страницы. Он легко настраивается и расширяется, и обладает обширной документацией и активным сообществом разработчиков.
2. Foundation ‑ это гибкий CSS-фреймворк, который также предоставляет набор готовых компонентов и сеточную систему. Он изначально разработан для создания адаптивных и мобильных веб-приложений. Foundation обладает модульной архитектурой, которая позволяет разработчикам выбирать и использовать только необходимые компоненты. Он также предлагает набор гибких тем и инструментов для настройки внешнего вида и поведения компонентов.
3. Bulma ‑ легкий и простой в использовании CSS-фреймворк. Он предоставляет готовые классы для создания различных компонентов и макетов, таких как колонки, навигация, формы и другие элементы интерфейса. Bulma обладает простым и интуитивным синтаксисом, а также хорошо настраивается с помощью переменных Sass. Он не требует JavaScript для работы, что может быть преимуществом в некоторых случаях.
4. Material-UI ‑ это CSS-фреймворк, основанный на принципах дизайна Material Design от Google. Он предоставляет готовые компоненты и стили, соответствующие спецификации Material Design. Material-UI также обладает возможностями для создания анимации и эффектов. Он интегрируется хорошо с React, но также может использоваться и с другими фреймворками или библиотеками JavaScript. Material-UI предлагает множество настраиваемых тем и стилей, а также инструменты для создания собственных компонентов.

|  |  |  |
| --- | --- | --- |
| **Фреймворк** | **Плюсы** | **Минусы** |
| **Bootstrap** | Огромное сообщество и обширная документация; Большой набор готовых компонентов и стилей; Гибкая сеточная система; Легкая настройка. | Шаблонность; Большой размер и сложность. |
| **Foundation** | Гибкость и модульность; Адаптивность; Мощная сеточная система. | Мало готовых компонентов; Сложная настройка. |
| **Bulma** | Простота использования; Интуитивный синтаксис; Настраивается с помощью Sass; Не требует JavaScript. | Мало готовых компонентов; Относительно небольшое сообщество. |
| **Material-UI** | Основан на Material Design; Широкий набор готовых компонентов; Хорошая поддержка React и других JS-фреймворков; Множество настраиваемых тем и стилей. | Специфичность; Относительно сложная настройка. |

1. Анализ менеджеров состояний

Менеджер состояний (State Manager) в веб-приложениях ‑ это инструмент или библиотека, которая помогает организовать и управлять состоянием данных приложения. Состояние обычно включает в себя информацию, которая может изменяться в процессе работы приложения, такую как пользовательский ввод, получение данных с сервера или изменение состояния компонентов интерфейса.

Менеджеры состояний обеспечивают централизованное хранение и управление состоянием приложения, что упрощает синхронизацию данных между различными компонентами. Они предлагают способы для обновления, доступа и отслеживания состояния, а также обрабатывают изменения состояния и обновление интерфейса при необходимости.

Некоторые из популярных менеджеров состояний в веб-разработке включают:

1. Redux ‑ это популярный менеджер состояний для JavaScript-приложений. Он работает на основе принципа однонаправленного потока данных и централизованного хранилища состояния. Redux использует концепцию «actions» (действия), «reducers» (редьюсеры) и «store» (хранилище) для управления состоянием и обновлением интерфейса.
2. MobX ‑ менеджер состояний, который обеспечивает реактивное управление состоянием. Он использует наблюдение (observables) и реакции (reactions) для автоматического отслеживания изменений состояния и обновления соответствующих компонентов. MobX обладает простым синтаксисом и удобным API.
3. Vuex ‑ менеджер состояний, разработанный специально для фреймворка Vue.js. Он предлагает централизованное хранилище состояния и API для управления состоянием и его мутаций. Vuex также интегрируется хорошо с остальными функциями Vue.js, такими как система компонентов и реактивное обновление интерфейса.
4. Recoil ‑ это библиотека для управления состоянием, разработанная Facebook. Она предоставляет простой и декларативный способ определения и использования атомов (atoms) - независимых единиц состояния. Recoil обеспечивает автоматическую отслеживаемость зависимостей и эффективное обновление только необходимых компонентов при изменении состояния.
5. Языки программирования Back-end

Для разработки серверной части веб-приложения существует множество языков программирования, и выбор зависит от ваших потребностей, предпочтений и требований проекта. Некоторые популярные языки программирования для разработки серверной части веб-приложений включают:

1. JavaScript ‑ язык программирования, который может быть использован как для разработки клиентской, так и серверной частей веб-приложения. Для серверной разработки на JavaScript используются фреймворки и среды выполнения, такие как Node.js, Express.js, Nest.js и другие.
2. Python ‑ простой и популярный язык программирования, который имеет широкие возможности для разработки серверных приложений. Для серверной разработки на Python широко используются различные фреймворки, такие как Django, Flask, Pyramid и Tornado.
3. Ruby ‑ язык программирования, который известен своим фреймворком Ruby on Rails. Он предоставляет удобные инструменты для разработки серверных приложений.
4. Java ‑ популярный и мощный язык программирования, используемый для разработки масштабных серверных приложений. Для веб-разработки на Java широко применяются фреймворки, такие как Spring и JavaServer Faces (JSF).
5. C# ‑ язык программирования, разработанный компанией Microsoft, который активно используется для разработки серверных приложений с помощью фреймворка ASP.NET и его веб-фреймворка ASP.NET Core.
6. PHP ‑ язык программирования, наиболее распространенный для разработки веб-приложений. Для серверной разработки на PHP часто используются такие фреймворки, как Laravel, Symfony и CodeIgniter.
7. Go ‑ компилируемый язык программирования, разработанный в компании Google, который обеспечивает высокую производительность при разработке серверных приложений. Он имеет встроенную поддержку конкурентности и широко используется для создания масштабируемых веб-сервисов.

Это только некоторые из популярных языков программирования для разработки серверной части веб-приложений. Важно учитывать требования вашего проекта, ваш уровень опыта и экспертизы, а также экосистему и инструменты, доступные для каждого языка, чтобы принять правильное решение при выборе языка программирования.

1. Анализ Back-end фреймворков

Разработку серверной части веб-приложений невозможно представить без использования фреймворков, которые существенно упрощают разработку и сокращают ее время. Данных фреймворков существует огромное множество, при этом все они имеют свои особенности и различные наборы инструментов и библиотек. Самыми популярными из них являются:

1. Spring ‑ мощный фреймворк на языке Java для разработки серверных приложений. Он предоставляет множество модулей, таких как Spring MVC, Spring Boot и Spring Data, для облегчения разработки и интеграции.
2. Node.js ‑ среда выполнения JavaScript и Express.js ‑ легкий и гибкий веб-фреймворк для Node.js. Позволяют разрабатывать мощные API и веб-приложения.
3. Ruby on Rails (RoR) ‑ полноценный фреймворк для разработки веб-приложений на языке Ruby. Он обеспечивает высокую продуктивность разработки и имеет широкий набор инструментов и конвенций.
4. Django ‑ мощный фреймворк на языке Python для разработки веб-приложений. Он обеспечивает высокую производительность, безопасность и включает в себя множество функций «из коробки».
5. Flask ‑ легковесный фреймворк на языке Python, который предоставляет минимальный набор инструментов для разработки веб-приложений. Он позволяет гибко настраивать приложение под свои нужды.
6. ASP.NET и ASP.NET Core ‑ фреймворки на языке C#, разработанные компанией Microsoft. Они предоставляют мощные инструменты для разработки веб-приложений и включают в себя поддержку MVC (Model-View-Controller) и API.
7. Laravel ‑ популярный фреймворк на языке PHP для разработки веб-приложений. Он предлагает элегантный синтаксис, расширяемость и широкий набор функций, таких как маршрутизация, работа с базами данных и аутентификация.
8. Nest.js ‑ фреймворк на языке JavaScript/TypeScript для разработки масштабируемых серверных приложений. Он базируется на Express.js и предлагает модульную архитектуру и использование декораторов
9. Анализ СУБД

При разработке веб-приложений часто требуется использовать Системы Управления Базами Данных (СУБД) для хранения и управления данными. Вот несколько популярных СУБД, которые широко применяются в веб-разработке:

1. MySQL ‑ одна из самых популярных открытых реляционных СУБД. Она обеспечивает высокую производительность, надежность и поддерживает множество функций, таких как транзакции, схемы безопасности и репликация.
2. PostgreSQL ‑ мощная открытая реляционная СУБД с широкими возможностями. Она обладает расширяемостью, поддержкой географических данных, полнотекстовым поиском и другими продвинутыми функциями.
3. MongoDB ‑ гибкая и масштабируемая документоориентированная СУБД, использующая формат BSON (расширенный JSON). Она позволяет хранить и обрабатывать документы без жесткой схемы и хорошо подходит для разработки современных веб-приложений.
4. SQLite ‑ это встраиваемая реляционная СУБД, которая хранит базу данных в одном файле. Она легкая в использовании и не требует отдельного сервера, поэтому она хорошо подходит для простых веб-приложений или разработки на ранних этапах проекта.
5. Microsoft SQL Server ‑ это коммерческая реляционная СУБД, разработанная Microsoft. Она обеспечивает высокую производительность, надежность и широкий набор инструментов для разработки и управления базами данных.
6. Oracle ‑ это мощная коммерческая реляционная СУБД, широко используемая в крупных предприятиях. Она обладает масштабируемостью, высокой производительностью и поддерживает широкий спектр функций для управления данными.

Выбор СУБД зависит от многих факторов, таких как требования к производительности, масштабируемости, тип данных и бюджет. Необходимо учитывать эти факторы и требования вашего веб-приложения, чтобы принять правильное решение при выборе СУБД.

1. Реализация клиентской части веб-приложения
2. Реализация серверной части веб-приложения

Заключение

Системы управления проектами являются невероятно важным инструментом при разработке продуктов. Они помогают планировать и отслеживать прогресс разработки, улучшают качество продукта, а также способны существенно сокращать все издержки.

Существуют различные методологии, на которых основаны данные системы. Каждая из них имеет свои преимущества и недостатки. Выбор методологии зависит от конкретных целей и задач компании, при этом часто используются их комбинации. Было проанализировано большинство популярных методологий, для ознакомления с предметной областью.

На основе методологий управления проектами в наше время разрабатывается большое количество различных систем. Были рассмотрены и проанализированы самые популярные из них. Некоторые из систем управления проектами являются простыми в использовании и универсальными для разных сфер разработки, другие же обладают большим количеством функций, заточенных под определенный тип продукта. Были выявлены слабые и сильные стороны каждой системы. Выделенные плюсы и минусы решений могут быть использованы для разработки собственной системы управления проектами и задачами.

Также были рассмотрены типы программного обеспечения, а также сделан вывод, что наиболее подходящими для разработки систем управления проектами являются веб-приложения. Такие приложения могут использоваться на различных устройствах, благодаря чему могут предоставлять быстрый доступ к системе.

Мир стремительно меняется, и компаниям необходимо подстраиваться под новые условия рынка. Системы управления проектами предоставляют компаниям такую возможность, поэтому их создание и использование является очень актуальной темой в наше время.

Список литературы

1. Ковалевская Н.А. Управление ресурсами на предприятии в контексте задач управления проектами. // Интеллектуальный и ресурсный потенциалы регионов: активизация и повышение эффективности использования. –2019. –C. 55-61.
2. Цыплов Е.А., Новиков В.А., Хайитов Х.О. Применение информационных систем управления проектами в повышении эффективности системы управления качеством. // Форум молодых ученых. –2019. –№10(38). –C. 556-560.
3. Гнедых В.А. Что такое корпоративная система управления проектами и зачем она нужна компании. // Финансовый университет при Правительстве РФ. –2015. –№4-4(48). –С. 57-61.
4. Методологии управления проектами: 12 популярных подходов [Электронный ресурс]. // Asana. URL : https://asana.com/ru/resources/project-management-methodologies (дата обращения: 28.04.23).
5. Лозгачева Т.М., Табекина О.А., Федотова О.В. Agile и научная организация труда: практика применения гибких методов в России. //Ученые записки крымского федерального университета. –2019. –№2. –C. 48-59.
6. Коул Р. Блистательный Agile. Гибкое управление проектами с помощью Agile, Scrum и Kanban. // Санкт-Петербург: Питер. –2019. –304 с.
7. Бахиркин М.В., Лукин В.Н. Как собрать кубик рубика: Waterfall или Agile в современных ИТ-проектах. // Материалы XXII международной конференции по вычислительной механике и современным прикладным программным системам. –2021. –C. 636-638.
8. Жук А.И. Управление задачами в Agile и Waterfall. // Студенческий. –2019. –№1-1(45) –C. 35-37.
9. Котляр Е.В. Система управления проектами Канбан. // Бизнес-образование в экономике знаний. –2020. –№ 1. –С. 57-60.
10. Плюснина Е.В., Орлова С.Н., Плеханов С.В. Практическое применение системы Kanban на разнопрофильных предприятиях. // Проблемы развития современного общества. –2022. –C. 134-138.
11. Scrum vs Waterfall vs Agile vs Lean vs Kanban [Электронный ресурс] // Visual Paradigm. URL : https://www.visual-paradigm.com/scrum/scrum-vs-waterfall-vs-agile-vs-lean-vs-kanban/ (дата обращения: 28.04.23).
12. Jira | Программное обеспечение для отслеживания задач и проектов [Электронный ресурс]. // Atlassian. URL : https://www.atlassian.com/ru/software/jira (дата обращения: 28.04.23).
13. Управление проектами команды с любого устройства | Trello [Электронный ресурс]. // Trello. URL : https://trello.com (дата обращения: 28.04.23).
14. Управляйте работой, проектами и задачами своего коллектива в сети [Электронный ресурс]. // Asana. URL : https://asana.com (дата обращения: 28.04.23).