

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РТФ

ОТЧЕТ

о проектной работе

* по теме: Web-разработка
* по дисциплине: Проектный практикум 1 курс

Команда: Это временное название

Тимлид: Шишлянников Александр Юрьевич РИ-131002

Аналитик: Маклецов Владимир Владимирович

РИ-130943

Дизайнер: Умяров Александр Дмитриевич

РИ-130949

Фронтенд-разработчик: Курбанов Сулаймон Талатович

РИ-130933

Бэкенд-разработчик: Ахмадалиев Савелий Батырбекович

РИ-130933

Бэкенд-разработчик: Вахромеев Василий Алексеевич

РИ-130914

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc165369900)

[1. Целевая аудитория 4](#_Toc165369901)

[2. Определение проблемы 6](#_Toc165369902)

[3. Подходы к решению проблемы 9](#_Toc165369903)

[4. Анализ аналогов 11](#_Toc165369904)

[5. Календарный план проекта 12](#_Toc165369905)

[6. Сценарии использования 14](#_Toc165369906)

[7. Требования к продукту и к MVP 15](#_Toc165369907)

[8. Стек для разработки 17](#_Toc165369908)

[9. Прототипирование 19](#_Toc165369909)

[10. Проектирование и разработка системы 21](#_Toc165369910)

[ЗАКЛЮЧЕНИЕ 22](#_Toc165369911)

[СПИСОК ЛИТЕРАТУРЫ 23](#_Toc165369912)

[ПРИЛОЖЕНИЕ А 26](#_Toc165369913)

ВВЕДЕНИЕ

Представим ситуацию: первокурсник попадает в вуз и знакомится с понятием «Индивидуальная траектория студента» (далее – ИОТ). Ему представляется возможность самостоятельно выбрать желаемое время для посещения лекций и практик, а также выбор онлайн-курсов от разных институтов. В такой ситуации от предложенных вариантов могут разбежаться глаза, и студент может проставить ошибочно пары, ввиду невнимательности, а где-то у него произойдут накладки с предметами и расписание не составится нужным образом.

Вообще, эта проблема может возникнуть и у старших курсов, ведь предметов на выбор несколько, а вариантов расписания становится очень много. Как же быть? Можно заранее прописывать желаемое в блокнот, или, возможно, составить эксель таблицу с указанными парами, или можно работать с модеусом и выбирать нужное там. Варианты рабочие, но не всегда удобные, и самое главное – не автоматизированы от слова «совсем». Все приходится делать «ручками». Вот бы был продукт, который мог взять на себя самостоятельную обработку данных…

Чтобы решить данную проблему, наша команда поставила цель -создание продукта, основной возможностью которого, является подбор нужного расписания в соответствии с параметрами выбора пар, заданной специальности, курса. Также предполагается добавление важной информации о преподавателях в специальные страницы веб-приложения.

Для решения проблемы мы поставили следующие основные задачи:

1. Составить план работы со всеми задачами и исполнителями
2. Определить и изучить нашу ЦА
3. Разработать веб-приложение с парсером
4. Разработать компактный дизайн и внедрить его на сайт
5. Провести тестирование готового продукта на настоящих данных
6. Целевая аудитория

Для начала нужно было определить, что конкретно будет в продукте, кто будет пользоваться нашим продуктом, какие потребности закрывает продукт, когда он востребован для пользователя, почему наш продукт будет использоваться, где можно будет его использовать. Для определения целевой аудитории мы использовали методику 5W Марка Шеррингтона. Это наиболее распространенный способ определения целевой аудитории и психологических характеристик, которыми обладают потенциальные потребители.

Сегментация рынка произошла по 5 вопросам:

1. Что? (What?) – сайт, где есть выборка вариантов расписания, с дополнительными понятными условиями, чтобы каждый студент мог подобрать себе хорошее расписание и ничего не упустил
2. Кто? (Who?) – базы данных рассчитаны на пользователей 1 и 2 курсов ИРИТ-РТФ
3. Почему? (Why?) – автоматизация выбора расписания, предложение новых вариантов для составления, полная систематизация, удобство
4. Когда? (When?) – 2 раза в год, когда происходит выбор расписания на осенний и весенний семестры
5. Где? (Where?) – на нашем сайте, в любое время в любом месте(чаще всего в сентябре и феврале)



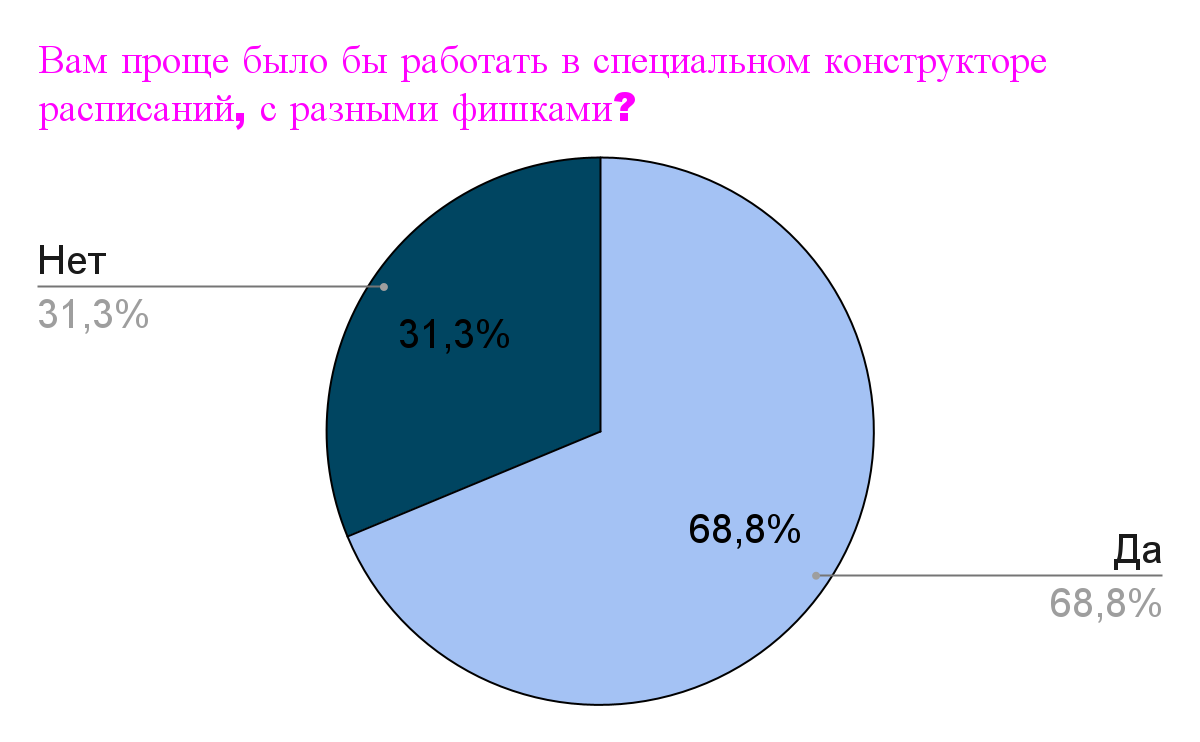
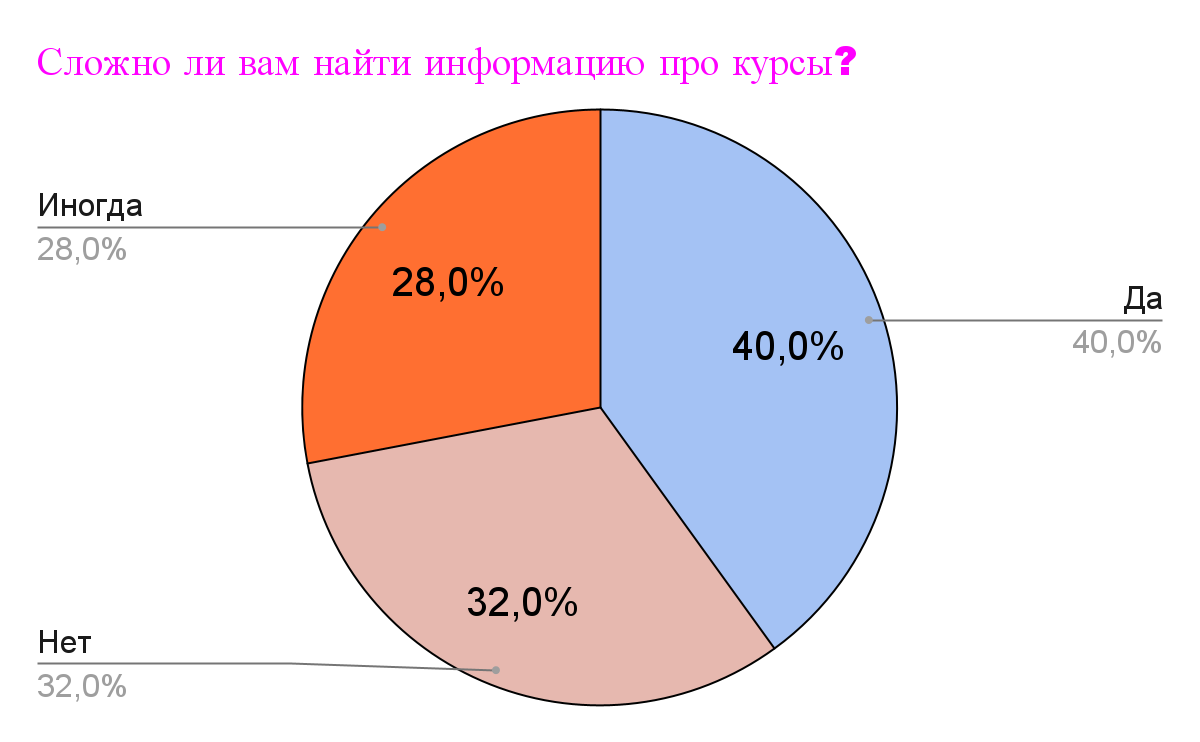
  

Рисунок 1-6. Аналитика в круговых диаграммах

1. Определение проблемы

Мы поставили проблему в отсутствии удобного и автоматизированного варианта для подбора вариантов расписания. Но думают ли также наши потенциальные пользователи? Как видно из изображений 3-6, согласно опросу, который мы провели среди студентов 1 и 2 курса ИРИТ-РТФ, такое веб-приложение будет востребовано. Есть трудности в сборке расписаний, трудно найти информацию про курсы.

У нашей проблемы есть нюансы. Она распространена на очень узкий круг пользователей, то есть – студентов определенного вуза определённых факультетов.

Почему проблема актуальна? Потому что она занимает много времени и сил, а также не всегда получается составить, то что хотел, из-за банальной невнимательности или неразборчивости.

1. Подходы к решению проблемы

Старайтесь предлагать как можно больше идей альтернативных решений – лучше откинуть неподходящее решение на этапе анализа, чем упустить выгоду из-за малого количества рассмотренных вариантов. Для этого предлагается проводить «мозговые штурмы» с проектной командой, интервьюировать заказчиков и пользователей и использовать методики ТРИЗ (Теория Решения Изобретательских Задач). Тщательное изучение проблемы, критериев и целей является эффективным началом в выявлении альтернативных решений.

Подготовка нескольких альтернатив на ранних стадиях процесса позволяет повысить вероятность принятия адекватного решения с четким пониманием всех возможных последствий и рисков.

Важно отметить, что зачастую для выбора окончательного решения может понадобиться несколько итераций. На каждой итерации мы получаем все больше информации об анализируемой проблеме и лучше понимаем факторы, влияющие на принятие решения. При этом можно уточнять и изменять аналитические параметры (например, критерии и их веса).

Результат анализа покажет, насколько каждый из выбранных критериев реализован в каждой из рассмотренных альтернатив.

Очевидно, что при использовании структурированного метода анализа предпочтение отдается тому решению, которое наберет наибольшее количество очков по шкале соответствия выбранным критериям. Однако возможны ситуации, при которых результаты «взвешивания» нескольких решений могут оказаться очень близкими. В этом случае стоит провести дополнительный анализ, например, вовлекая экспертов и представителей заинтересованных сторон, выявляя и добавляя новые критерии выбора.

*Например*

*Чтобы сократить процесс согласования счетов, мы можем:*

1. Перераспределить задачи между согласующими. В результате несколько человек могут быть исключены из процесса. Суммарное время процесса сократится за счет периодов передачи данных/ожидания/коммуникации при передаче.
2. Перейти на электронный документооборот — достоверность счетов и данных в них будет подтверждена оператором ЭДО, подтверждение человеком не потребуется.
3. Автоматически распознавать сканы счетов и сравнивать данные с цифрами из системы закупок. Ручная проверка и согласование не потребуются.

Чтобы продумать все варианты, надо разобраться – а что же происходит сейчас? Как устроен процесс без вашей системы, как работают пользователи и заказчики? Даже если процесса еще нет, подробная информация про текущее состояние очень важна. Так мы поймем, какое решение устранит проблему, а не создаст еще одну.

1. Анализ аналогов

Прямые конкуренты – это аналогичное программное обеспечение на аналогичном рынке и работающий с целевой аудиторией, что и разрабатываемый продукт. Косвенные конкуренты – это похожее программное обеспечение с другими характеристиками или абсолютно другой продукт, но работающие с целевой аудиторией, что и разрабатываемый продукт.

Как правильно провести анализ конкурентов?

Рассмотрим шесть базовых шагов, которые надо выполнить при проведении подобной аналитики.

* определить цели анализа конкурентов;
* определить конкурентов;
* определить критерии анализа конкурентов;
* приступить к анализу (заполнение таблиц);
* сопоставить полученные данные;
* сделать выводы.

Используйте аналогичный продукты для того, чтобы заимствовать наиболее удачные приемы и способы решения проблем.

В таблице 2 представлен пример анализа конкурентов по трем критериям.

Таблица 2 – Анализ конкурентов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии сравнения | Разделим чек | Splitwise | Делить чек | Скинемся! |
| Многопользовательский режим | нет | да | да | да |
| Возможность сканировать фото | нет | нет | да | нет |
| Отдельное приложение | да | да | да | нет |

1. Календарный план проекта

Название проекта:

Руководитель проекта:

Таблица 1 – Календарный план

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Работы** | **Ответственный** | **Длительность** | **Дата начала** | **Временные рамки проекта** | | | | | | | |
| 1 нед | 2 нед | 3 нед | 4 нед | 5 нед | 6 нед | 7 нед | 8 нед |
| **Анализ (08.04.2024 - 14.04.2024)** | | | | | | | | | | | | |
| *1.1* | *Провести аналитику, определить ЦА, определить ценность* | Аналитик | 1 неделя | 01.04.2024 |  |  |  |  |  |  |  |  |
| *1.2* | *Use Case* | C#-разработчик - логика, ТимЛидер | 1 неделя | 01.04.2024 |  |  |  |  |  |  |  |  |
| *1.3* | *Написать MVP* | ТимЛидер | 1 неделя | 01.04.2024 |  |  |  |  |  |  |  |  |
| *1.4* | *Написать скелет веб-приложения и опубликовать* | C#-разработчик - логика | 2 недели | 01.04.2024 |  |  |  |  |  |  |  |  |
| *1.5* | *Представить как должен выглядеть веб-приложение* | Фронт - энд разработчик, Дизайнер | 2 недели | 01.04.2024 |  |  |  |  |  |  |  |  |
| **Проектирование (15.04.2023 - 21.04.2024)** | |  |  |  |  |  |  |  |  |  |  |  |
| *2.1* | *Написать модели веб-приложения* | C#-разработчик - логика | 1 неделя | 15.04.2024 |  |  |  |  |  |  |  |  |
| *2.2* | *Создать прототип парсера* | C#-разработчик - парсер | 1 неделя | 15.04.2024 |  |  |  |  |  |  |  |  |
| *2.3* | *Реализация дизайна главной страницы(текст на главной странице)* | Дизайнер | 1 неделя | 15.04.2024 |  |  |  |  |  |  |  |  |
| *2.4* | *Первая реализация главной страницы(?)* | Фронт - энд разработчик | 1 неделя | 15.04.2024 |  |  |  |  |  |  |  |  |
| **Разработка (22.04.2024 - 19.05.2024)** | |  |  |  |  |  |  |  |  |  |  |  |
| *3.1* | *Реализация дизайна веб-приложения\** | Фронт - энд разработчик | 2 недели | 22.04.2024 |  |  |  |  |  |  |  |  |
| *3.2* | *Создание идей для дизайна\** | Дизайнер | 2 недели | 22.04.2024 |  |  |  |  |  |  |  |  |
| *3.3* | *Создать парсер\** | C#-разработчик - парсер | 2 недели | 22.04.2024 |  |  |  |  |  |  |  |  |
| *3.4* | *Реализация страницы логирования* | C#-разработчик - логика | 2 недели | 22.04.2024 |  |  |  |  |  |  |  |  |
| *3.5* | *Создание тест - кейсов* | ТимЛидер | 2 недели | 22.04.2024 |  |  |  |  |  |  |  |  |
| *3.6* | *Реализовать коструктор расписаний\** | C#-разработчик - логика | 3 недели | 29.04.2024 |  |  |  |  |  |  |  |  |
| *3.7* | Добавление фич | C#-разработчики | 1 неделя | 13.05.2024 |  |  |  |  |  |  |  |  |
| **Защита (20.05.2024 - 22.06.2024)** | |  |  |  |  |  |  |  |  |  |  |  |
| *4.1* | *Сделать презентацию* | Аналитик/ТимЛидер | 1 неделя | 20.05.2024 |  |  |  |  |  |  |  |  |
| *4.2* | *Защитить проект* | Все | 1 неделя | 22.06.2024 |  |  |  |  |  |  |  |  |

1. Сценарии использования

Сценарий использования (use case) — описание поведения системы, когда она взаимодействует с кем-то (или чем-то) из внешней среды. Сценарий использования описывает, «кто» и «что» может сделать с рассматриваемой системой, или что система может сделать с «кем» или «чем». Методика сценариев использования применяется для выявления требований к поведению системы, известных также как пользовательские и функциональные требования.

Каждый сценарий использования сосредотачивается на описании того, как достигнуть цели или задачи. Для большинства программных проектов это означает, что потребуется множество сценариев использования, чтобы определить необходимый набор свойств новой системы.

Сценарий использования определяет взаимодействия между внешними агентами и системой, направленные на достижение цели. Актор представляет собой роль, которую играет человек или вещь, взаимодействуя с системой. Один и тот же человек, использующий систему, может быть представлен как различные акторы, потому что они играют различные роли.

Сценарии использования рассматривают систему как «черный ящик», и взаимодействия с системой, включая системные ответы, описываются с точки зрения внешнего наблюдателя.

Сценарий использования должен:

* описывать, что именно система должна сделать, чтобы актор достиг своей цели;
* не затрагивать деталей реализации;
* иметь достаточный уровень детализации.

1. Требования к продукту и к MVP

Требования систематизируются несколькими способами.

Требования клиентов

Клиенты – это те, кто выполняет основные функции системного проектирования, со специальным акцентом на пользователе системы как ключевом клиенте. Пользовательские требования определят главную цель системы.

Функциональные требования

Функциональные требования объясняют, что должно быть сделано. Они идентифицируют задачи или действия, которые должны быть выполнены. Функциональные требования определяют действия, которые система должна быть способной выполнить, связь входа/выхода в поведении системы.

Нефункциональные требования

Нефункциональные требования – требования, определяющие свойства, которые система должна демонстрировать, или ограничения, которые она должна соблюдать, не относящиеся к поведению системы. Например, производительность, удобство сопровождения, расширяемость, надежность, факторы эксплуатации.

Производные требования

Требования, которые подразумеваются или преобразованы из высокоуровневого требования. Например, требование для большего радиуса действия или высокой скорости может привести к требованию низкого веса.

*Например*

*Анкета должна содержать файл с фото, так как фото необходимо при оформлении документов – это бизнес-требование. А возможно, и бизнес-правило.*

*Из бизнес-требования следует, что у пользователя должна быть возможность прикрепить фото к анкете – это пользовательское требование. То есть требование, описывающее действия пользователя.*

*Получается, что система должна иметь функционал прикрепления фото к анкете – это уже функциональное требование, описывающее поведение системы. Или как должна работать система, чтобы выполнять исходное пользовательское требование.*

*Будем хранить все фото в формате base64 в отдельной таблице в БД – это нефункциональные требования.*

*Фото в очень хорошем качестве нам не нужно, а также мы не хотим покупать много памяти для сервера. Поэтому сделаем ограничение на размер загружаемого фото: не более 10 Мб.*

На каждое бизнес-требование, как правило, приходится несколько пользовательских. Пользовательское требование декомпозируется на какое-то число функциональных. К каждому функциональному требованию нужно продумать нефункциональные требования и ограничения.

Также нефункциональные требования и ограничения могут напрямую вытекать как из пользовательских требований, так и из бизнес-требований и правил.

Таким образом получаются деревья из требований, в вершине каждого из которых – бизнес-требование.

1. Стек для разработки

ASP.NET Core – кроссплатформенная технология созданная компанией

Microsoft, для создания веб-сервисов, веб-приложений, сайтов.

Плюсы:

Разделение ответственности: технология ASP.NET состоит из 3

частей (паттерн MVC, M – model, V – view, C – controller). Подобная

архитектура позволит нам проще масштабировать наше приложение, тк мы

точно не знаем, всё ли мы успеем выполнить или реализуем только MVP, это

поможет нам, в случае если мы захотим расширить проект. Также

распределение ответственности в нашей команде добавляет плюс в копилку

MVC. Технология MVC позволяет нам работать параллельно, не завися от

того, когда и какое обновление выкатит другой разработчик, тк мы будем

работать в разных папках этого проекта. + тк над проектом работают новички

готовая архитектура поможет не нагородить дел с самого начала

Относительно легко поддерживается и тестируется

Гибкость: технология ASP.NET гибкая, мы можем настраивать

поведение без какого-либо внедрения в код, мы можем использовать готовые

компоненты, которые нам предоставляет Microsoft

C# — универсальный язык общего назначения!

1. Прототипирование

Процесс создания прототипа состоит из четырёх шагов:

* определение начальных требований;
* разработки первого варианта прототипа (в ПО. например, который содержит только пользовательский интерфейс системы);
* этап изучения прототипа заказчиком и конечным пользователем. Получение обратной связи о необходимых изменениях и дополнениях;
* переработка прототипа с учетом полученных замечаний и предложений.

*Качества, которыми должен* обладать *эффективный прототип*

Этап создания прототипа не должен быть затяжным.

Эффективные прототипы являются одноразовыми. Они предназначены для того чтобы донести идею до заинтересованного лица. После того как идея была донесена, прототип может быть отвергнут.

Эффективные прототипы являются сфокусированными, это означает что следует обращать внимание на сложные части при создании прототипов. Необходимо найти шаблонные взаимодействия, которые давно известны в теории юзабилити.

Необходимо обращать внимание на элементы взаимодействия, которые принесут пользу вашему продукту.

Даже самые опытные специалисты-разработчики допускают ошибки. Это правило особенно очевидно в коллективной работе. По мере выполнения проекта накапливаются мелкие ошибки, допущения, неверные решения. В результате получается плохой продукт при хорошей исходной идеи. Разработка прототипа – средство, позволяющее проанализировать идеи, прежде чем на них будут потрачены время и деньги. Это распространенный в инженерной практике метод. Основная цель, достигаемая при создании прототипа, – это экономия времени и ресурсов. Ценность прототипа заключается в том, что он является внешней оболочкой-моделью отражающей существенные отношения разрабатываемого продукта. По сравнению с реальным продуктом прототипы просты и недороги в разработке. При минимальном вложении средств можно обнаружить ошибки создателей и юзабилити проблемы, и улучшить пользовательский интерфейс до того, как сделаны значительные инвестиции в окончательную разработку и технологии.

Создание эффективного прототипа интерфейса является чрезвычайно важной задачей. Прототип должен хорошо выглядеть, чтобы понравиться заказчику и не вызвать вопросов у субъектов тестирования, он должен быть максимально дёшев, максимально полон и, что немаловажно, должен с лёгкостью обновляться.

Требования к прототипу изменяются со временем. Сначала наиболее актуальными его свойствами являются скорость создания и простота модификации. Эти свойства позволяют быстро разработать и проверить несколько версий интерфейса, при этом ещё и исправить значительную часть ошибок.

Затем на первый план выходят функциональность и эстетичность, простота же модификации уже не столь важна, поскольку с каждой новой исправленной ошибкой снижается вероятность того, что прототип придётся полностью переделывать при обнаружении новой ошибки.

Поэтому всегда правильно сделать прототип настолько похожим на результирующую систему, насколько позволяет самая поздняя его версия. Первый прототип стоит делать максимально примитивным. Только после того, как тестирование подтверждает его правильность, стоит сделать более детализированный прототип.

Прототипы интерфейсов возможно привести в приложении к отчету.

1. Проектирование и разработка системы

На этом этапе формируется общая структура программного комплекса. В соответствии с технологией нисходящего структурного программирования программный комплекс разбивается на небольшие части – программные модули (блоки). Для каждого программного модуля формулируются требования по реализуемым функциям и разрабатывается алгоритм, реализующий эти функции. Необходимо привести описание этих модулей, а также задач, которые они решают.

Необходимо привести алгоритм работы приложения, т.е. точного предписания и последовательности действий, заложенных в логике продукта.

Также на этом этапе можно привести схему взаимодействия программных модулей, т.е. схема потоков данных программы.

В этой части можно привести описание самого процесса разработки разнотипных сложных блоков программного обеспечения.

ЗАКЛЮЧЕНИЕ

Заключение должно содержать общие выводы, обобщенное изложение основных проблем. Могут быть указаны положительные и отрицательные стороны разработанного решения. Должны быть указаны перспективы дальнейшей разработки темы. Примерный объем заключения 2-3 страницы.

В заключительной части работы должно быть дано последовательное, логическое изложение полученных результатов в соответствие поставленными изначально задачами.

На данный момент сервис готовится к запуску и тестированию на реальных пользователях.

СПИСОК ЛИТЕРАТУРЫ

1. Веб-технологии для разработчиков Обзор протокола HTTP / Веб-технологии для разработчиков. – 2022. – URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Overview (дата обращения: 30.04.2022).
2. Вигерс К. Разработка требований к программному обеспечению / К. Вигерс, Дж. Битти. – 3-е изд. – Москва: Русская редакция, 2014. – 736 с. – ISBN 978-5-9909805-3-2.
3. Евлантьев С.Карта эмпатии клиента / Станислав Евлантьев. – 2022. – URL: https://in-scale.ru/blog/karta-empatii/ (дата обращения: 01.05.2022).
4. Ильяхов М. Пиши, сокращай: как создать сильный текст / М. Ильяхов, Л. Сарычева. – 3-е изд. – Москва: Альпина Паблишер, 2022. – 440 с. – ISBN 978-5-9614-6526-6.
5. Кинзябулатов Р. IDEF0. Знакомство с нотацией и пример использования / Рамиль Кинзябулатов. – 2022. – URL: https://trinion.org/blog/idef0-znakomstvo-s-notaciey-i-primer-ispolzovaniya (дата обращения: 30.04.2022).
6. Коберн А. Современные методы описания функциональных требований к системам / А. Коберн. – Москва: Издательство «Лори», 2012. – 264 с. – ISBN 978-5-85582-326-4.
7. Мидоус Ф. Введение в протоколы HTTP и HTTPS / Филлип Мидоус. – 2022. – URL: https://docs.microsoft.com/ru-ru/azure/rtos/netx-duo/netx-duo-web-http/chapter1 (дата обращения: 05.05.2022).
8. Сырцев В. Использования диаграммы вариантов использования UML при проектировании программного обеспечения / Вадим Сырцев. – 2021. – URL: https://habr.com/ru/post/566218/ (дата обращения:30.04.2022).
9. Фирстов Л. О развитии навыка говорения / Леонид Фирстов. – 2020. – URL: https://habr.com/ru/post/494424/ (дата обращения: 30.04.2022).
10. Фитцпатрик Р. Спроси маму: Как общаться с клиентами и подтвердить и подтвердить правоту своей бизнес-идеи, если все кругом врут? / Р. Фитцпатрик. – Москва : Альпина Паблишер, 2021. – 160 с. – ISBN 978-5-9614-3045-5.
11. Шейнблат В. Естественный подход к освоению языка / Владимир Шейнблат. – 2022. – URL: https://englishsimple.ru/articles/natural-approach-to-language-learning/ (дата обращения: 25.04.2022).
12. Шторкин С. Средневзвешенная система голосования / Семен Шторкин – 2022. – URL: https://habr.com/ru/post/63664/ (дата обращения: 30.04.2022).
13. Apple MacOS / Apple. – 2022. – URL: https://www.apple.com/za/macos/what-is/ (дата обращения: 30.04.2022).
14. Best Programmer Функциональные и нефункциональные требования: полное руководство / Best Programmer. – 2021. – URL: https://bestprogrammer.ru/izuchenie/funktsionalnye-i-nefunktsionalnye-trebovaniya-polnoe-rukovodstvo (дата обращения: 15.05.2022).
15. Google Chrome Браузер от Google / Google Chrome. – 2022. – URL: https://www.google.ru/chrome/ (дата обращения: 30.04.2022).
16. HeyLady Женское сообщество для создания конференций и практики английского языка / HeyLady. – 2022. – URL: https://www.heylady.io/ (дата обращения: 30.04.2022).
17. Lingbe Бесплатные звонки с носителями в любое время / Lingbe. – 2022. – URL: https://lingbe.com/ (дата обращения: 30.04.2022).
18. PostgrePro Что такое PostgreSQL? / PostgrePro. – 2020. – URL: https://postgrespro.ru/docs/postgresql/12/intro-whatis (дата обращения: 30.04.2022).
19. Proglib Об Agile / Denver. – 2022. – URL: https://proglib.io/p/klyuchevye-razlichiya-mezhdu-agile-scrum-i-kanban-2022-02-24 (дата обращения 05.04.2022).
20. SimilarWeb Аналитика трафика Tandem / SimilarWeb. – 2022. – URL: https://www.similarweb.com/ru/website/tandem.net/#traffic (дата обращения: 30.04.2022).
21. Tandem / Tandem.net. – 2022. – URL: https://www.tandem.net/ru (дата обращения: 30.04.2022).
22. JavaScript WebSocket / JavaScript. – 2022. – URL: https://learn.javascript.ru/websocket (дата обращения: 02.05.2022).
23. VC.RU Анализ конкурентов на практике / VC.RU. – 2020. – URL: https://vc.ru/marketing/181297-analiz-konkurentov-na-praktike-10-shagov-shablon (дата обращения: 30.04.2022).
24. VC.RU Как составить портрет клиента (целевой аудитории) / VC.RU. – 2020. – URL: https://vc.ru/marketing/156147-kak-sostavit-portret-klienta-celevoy-auditorii-instrukciya-s-primerami (дата обращения: 30.04.2022).

ПРИЛОЖЕНИЕ А

