Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Формирование требований к информационной системе «Сервис для раздельного сбора отходов EcoSort»

Студент: Трубач Д. С.

ФИТ 4 курс 5 группа

Преподаватель: Якубенко К. Д.

Минск 2024

# Введение

В современном мире, с увеличением объёмов отходов и повышением внимания к экологическим проблемам, создание веб-приложения для сортировки мусора становится всё более актуальным и важным. На территории РБ не так хорошо развита сортировка отходов. Решением данной проблемы может стать программное обеспечение, которое поможет разобраться как начать раздельный сбор, заинтересует системой поощрения.

Целью технического задания является построение информационной модели и на её основе формирование пользовательских и системных требований. Для достижения этой цели необходимо решить следующие задачи: разработать удобный и интуитивный пользовательский интерфейс, обеспечивающий простую навигацию и доступ к информации о правилах сортировки отходов, предоставить пользователям функционал просмотра руководств как правильно производить сортировку,а также разработка бонусной системы баллов здля дальнейшего их обмена на продукцию, произведенную из переработанных отходов.

Первоначально программное средство предназначено для жителей города Минск, в дальнейшем возможно развитие в других населенных пунктах.

В использовании приложением нет строгих ограничений. Важным является тот факт, что пользователь должен полностью понимать, как использовать информацию из приложения и производить обмен баллов на продукцию могут люди, которые легко могут добраться до пункта обмена.

Основные задачи приложения:

* разработать удобный интерфейс, который позволит пользователям легко ориентироваться и находить информацию о сортировке отходов;
* предоставить функционал для ознакомления с пошаговыми инструкциями по раздельному сбору мусора;
* создать систему накопления баллов за сортировку, с возможностью их обмена на товары, произведенные из переработанных материалов.

# 1 Назначение разработки

Разработка приложения, посвященному раздельному сбору мусора, имеет следующие ключевые цели:

* функциональность: страницы должны загружаться быстро, навигация должна быть удобной, а интерфейс - простым и понятным;
* информативность: на сайте должна быть полная и понятная информация об услугах. Клиент должен найти все ответы на свои вопросы на сайте;
* ориентированность на клиента: сайт должен содержать актуальную информацию о предлагаемых услугах, поиск информации должен быть быстрым, а обратная связь – удобной;
* визуальная привлекательность: внешний вид сайта играет важную роль, так как пользователь сначала оценивает его визуально. Чтобы привлечь клиента, сайт должен быть привлекательным;
* безопасность: клиент должен чувствовать себя защищенным и доверять ресурсу.

Целью разработки веб-приложения для сортировки мусора является предоставление удобного и полезного инструмента для всех пользователей, включая частных лиц и организации. Приложение будет стремиться удовлетворить потребности каждой группы, предлагая простой доступ к информации о правилах сортировки, систему мотивации в виде бонусов за правильную утилизацию, а также полезные рекомендации и советы. Важными аспектами разработки станут безопасность данных, высокая производительность, визуальная привлекательность интерфейса, а также применение современных технологий для достижения экологических целей.

Разработка такого приложения требует комплексного подхода, включая адаптацию под мобильные устройства, обеспечение поддержки пользователей через чат или горячую линию, а также регулярное обновление контента в зависимости от изменений в экологических нормах и правилах. В конечном итоге, приложение должно стать не только полезным инструментом, но и платформой, объединяющей общество вокруг общей цели — сохранения окружающей среды и сокращения отходов.

# 2 Основание для разработки

Наименованием веб-приложения по раздельному сбору вторсырья является «EcoSort». Веб-приложение «EcoSort» разрабатывается в качестве дипломного проекта для получения квалификации «Инженер-программист» по специальности «Программное обеспечение информационных технологий» на Факультете Информационных Технологий Белорусского Государственного Технологического Университета. На основании стандарта организации подготовки, представления и защиты дипломных работ СТП БГТУ 001-2019, утвержден и введён в действие приказом № 108 от 20.03.2019 Министерством образования Республики Беларусь.

В рамках данного дипломного проекта акцент будет сделан на применении современных технологий веб-разработки и интеграции передовых решений для создания удобного и функционального продукта. Основная цель разработки – продемонстрировать владение широким спектром навыков, необходимых для создания профессионального программного обеспечения, включая проектирование архитектуры системы, разработку интерфейса, а также обеспечение безопасности и масштабируемости приложения.

Выполнение проекта требует глубокого анализа существующих решений на рынке, понимания требований пользователей и внедрения лучших практик в разработке веб-приложений. Особое внимание будет уделено качеству кода, соответствию требованиям производительности и удобству использования системы. Все этапы разработки будут документированы в соответствии с требованиями СТП БГТУ 001-2019, включая проведение тестирования и демонстрацию работоспособности готового продукта.

Таким образом, данный проект направлен на создание инновационного и эффективного веб-приложения, которое не только отвечает актуальным требованиям рынка программного обеспечения, но и демонстрирует профессиональные навыки и знания выпускника в области информационных технологий.

В пояснительной записке будет содержатся информация о сопоставимых продуктах, структуре и реализации проекта, тестирование веб-приложения, а также инструкции по использованию приложения.

# 3 Требования к программному изделию

# 3.1 Требования к функциональным характеристикам

Разрабатываемое программное средство «EcoSort» должно представлять собой web-приложение, разделенное на две составляющие: серверная и клиентская части.

Взаимодействие с клиентской часть приложение осуществляется путем использования браузера, установленного на персональный компьютер.

В системе должен быть обеспечен надлежащий уровень защиты информации.

Разрабатываемое приложение должно иметь удобный и интуитивно понятный интерфейс.

Надежное и устойчивое функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечисленных ниже:

* организация бесперебойного электрического питание;
* защита персональных данных пользователей паролем, которые хранятся в захешированом виде;
* выполнением требований «ГОСТ 31078-2002. Защита информации. Испытания программных средств на наличие компьютерных вирусов»;
* высокая квалификация и должный инструктаж персонала, взаимодействующего с информационной системой.

Необходимо обеспечить максимально безотказную работу программы при любых входных данных, а также при любых действиях пользователя.

# 3.2 Требования к надежности

Программное средство для выполнения требований надежности должно:

* хранить пароль пользователя в зашифрованном виде;
* ограничить возможность создания более одной учетной записи на одну электронную почту;
* передавать данные по зашифрованному протоколу HTTPS;
* обеспечивать валидацию введенных пользователем данных;
* обеспечивать защиту личных данных пользователей;
* обеспечивать идентификацию, аутентификацию и авторизацию пользователей.

Для выполнения требований надежности программное средство должно включать в себя ряд ключевых механизмов и технологий, обеспечивающих безопасность и защиту данных пользователей.

Во-первых, хранение паролей пользователей должно осуществляться в зашифрованном виде. Для этого используются алгоритмы хэширования, такие как **bcrypt** или **argon2**, которые гарантируют, что даже в случае утечки базы данных пароли останутся защищенными и не будут доступны злоумышленникам в оригинальном виде. Важно также использовать **соль** (случайные данные, добавляемые к паролю перед хэшированием), что предотвращает использование радужных таблиц для дешифровки паролей.

Во-вторых, система должна ограничивать возможность создания более одной учетной записи на одну электронную почту. Это поможет предотвратить злоупотребления и создать уникальные идентификаторы для каждого пользователя.

Передача данных между клиентом и сервером должна осуществляться исключительно по зашифрованному протоколу **HTTPS**. Этот протокол использует **TLS** (Transport Layer Security) для шифрования данных, передаваемых через сеть, что предотвращает их перехват третьими лицами. Кроме того, необходимо убедиться, что сертификаты безопасности сервера регулярно обновляются и проверяются на валидность.

# 3.3 Условия эксплуатации

Приложение должно быть спроецировано таким образом, чтобы предоставить возможность для комфортного использования всеми членами целевой аудитории – то есть людьми всех полов, возрастов и социальных положений. Для достижения этого, интерфейс приложения должен быть интуитивно понятным и удобным в использовании.

# 3.4 Требования к составу и параметрам технических средств

Минимальные аппаратные требования к пользовательской машине:

* процессор архитектуры x86 или arm;
* минимальное количество ядер процессора – 2;
* минимальный объем оперативной памяти – 4 Гб;
* минимальный доступный объем дискового пространства – 20 Гб.

Рекомендуемые аппаратные требования к пользовательской машине:

* минимальное количество ядер процессора – 4;
* минимальный объем оперативной памяти – 8 Гб;
* минимальный доступный объем дискового пространства – 40 Гб.

# 3.5 Требования к информационной и программной совместимости

Язык программирования, используемый серверной частью – C#.

Платформа для написания WebSocket-сервера – NodeJS.

Язык программирования, используемый клиентской частью – JavaScript с использованием React.

Исходный язык, используемый для написания структурированных запросов к базе данных – MySQL.

Передача данных осуществляется по протоколу HTTP. Связь между сервером и СУБД – локальная.

# 3.6 Требования к программной документации

Программная документация должна быть представлена в виде отдельной конечной точки приложения как для клиентской, так и для серверной части. Она должна быть доступна для разработчиков и других участников проекта на всех этапах жизненного цикла приложения, предоставляя подробное описание функционала и API для его эффективного использования и интеграции. Помимо этого, внутри кода программы должны быть предусмотрены комментарии, которые помогут разработчикам и пользователям глубже понимать логику и структуру кода, облегчат сопровождение проекта и внесение изменений.

Документация для серверной части должна быть организована в формате описания всех конечных точек (API). Это описание должно включать следующие элементы:

* URL конечной точки;
* название используемого HTTP метода;
* пример данных, передаваемых на сервер по указанной конечной точке;
* пример данных, возвращаемых с сервера.

Для достижения этих целей может быть использована спецификация описания API под названием OpenAPI. Она предоставляет стандартизированный формат для документирования RESTful API и может быть интегрирована с инструментами, такими как Swagger, для автоматической генерации интерактивной документации. Это упростит тестирование API и взаимодействие с ним для разработчиков, тестировщиков и других пользователей приложения.

Документация также должна быть интегрирована в процесс непрерывной интеграции и доставки (CI/CD), чтобы гарантировать её актуальность и соответствие текущей версии приложения. Это можно реализовать с помощью автоматических проверок на этапе сборки приложения, которые будут отслеживать изменения в API и автоматически обновлять документацию. Таким образом, каждый новый релиз приложения будет сопровождаться обновленной документацией, что позволит избежать устаревших данных и облегчить разработчикам и тестировщикам работу с новыми версиями.

# 4. Технико-экономические требования

Разработка и эксплуатация веб-приложения для сортировки отходов должна быть оптимизирована с точки зрения технических и экономических показателей для достижения эффективности и минимизации затрат.

**Производительность и масштабируемость**. Программное обеспечение должно быть спроектировано таким образом, чтобы поддерживать растущее количество пользователей и обеспечивать стабильную работу даже при высокой нагрузке. Для этого требуется использование масштабируемых облачных решений, таких как **AWS**, **Microsoft Azure** или **Google Cloud**, которые позволяют динамически распределять ресурсы и обеспечивать высокую доступность сервиса. Вертикальное и горизонтальное масштабирование должно быть заложено на уровне архитектуры.

**Оптимизация затрат на инфраструктуру**. Для минимизации затрат на поддержание серверов и инфраструктуры следует использовать модель **pay-as-you-go**, предоставляемую облачными провайдерами, которая предполагает оплату только за реально использованные ресурсы (процессорное время, память, трафик). Это позволит избегать лишних расходов при низких нагрузках и гибко управлять бюджетом в периоды пикового использования.

**Экономия ресурсов на этапе разработки**. Для сокращения времени и стоимости разработки рекомендуется использовать готовые библиотеки и фреймворки с открытым исходным кодом, а также инструменты автоматизации, такие как CI/CD (Continuous Integration/Continuous Deployment) для ускорения разработки, тестирования и развертывания приложения. Это также снизит требования к численности команды разработчиков.

**Минимизация расходов на техническую поддержку**. Внедрение системы мониторинга и автоматической диагностики проблем позволит значительно снизить затраты на техническую поддержку и оперативное решение инцидентов. Система должна включать инструменты для отслеживания метрик производительности, а также автоматизированные уведомления о возможных сбоях или аномалиях в работе приложения.

**Энергоэффективность**. Программное обеспечение должно быть оптимизировано для снижения нагрузки на серверы и минимизации энергопотребления. Это может быть достигнуто путем применения технологий **кэширования данных**, эффективного управления потоками и ресурсами, а также снижения использования серверных мощностей при низкой активности пользователей.

**Юридические и лицензионные аспекты**. Для минимизации расходов на лицензирование следует использовать свободное программное обеспечение (open-source) и соответствующие технологии. Кроме того, необходимо учитывать соответствие законодательству о защите данных (например, GDPR), чтобы избежать возможных штрафов за нарушение прав пользователей на конфиденциальность.

# 5 Стадии и этапы разработки

Разработка должна быть проведена в 3 этапа:

* техническое задание;
* технический проект;
* внедрение.

На стадии «Техническое задание» важным этапом является его разработка, согласование и утверждение. Этот процесс должны включать в себя следующие важные шаги:

1. **Разработка**: на основе бизнес-требований и анализа функционала разрабатывается первоначальный вариант технического задания. Важно учитывать все детали проекта, включая архитектурные решения, спецификации, функциональные и нефункциональные требования.
2. **Согласование**: после разработки ТЗ отправляется на согласование с заинтересованными сторонами: заказчиками, техническими специалистами, юристами, если это необходимо. В процессе согласования могут быть выявлены недочеты или предложения по улучшению, которые необходимо внести в документ.
3. **Утверждение**: после завершения всех корректировок и доработок техническое задание утверждается официально.

Таким образом, утвержденное техническое задание становится основным документом, на который опираются все участники проекта на последующих этапах разработки.

На стадии «Технический (и рабочий) проект» должны быть выполнены перечисленные ниже этапы работ:

* разработка программы;
* разработка программной документации;
* испытание программы.

На этапе разработки программы должна быть выполнена работа по программированию и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77.

# 6 Описание информационной вербальной модели

Объекты информационной вербальной модели для приложения могут быть представлены следующим образом:

Описание объекта «Пользователь»:

* Описание: физическое или юридическое лицо, которое использует приложение для обмена информацией о вторсырье, накопления баллов и получения скидок.
* Атрибуты: имя, адрес электронной почты, количество накопленных баллов, роль.
* Связан с Статьями (публикует статьи), Рейтингами (оставляет комментарии к статьям), Лайками (ставит оценку статьям),

Описание объекта «Статья»:

* Описание: публикация в приложении, содержащая информацию, связанную со вторсырьем или другим контентом.
* Атрибуты: название, текст, дата публикации, ссылка на изображение, количество лайков.
* Связано с Пользователем, Рейтингом и Лайками (пользователь публикует статью, в свою очередь, статья имеет комментарии и лайки).

Описание объекта «Пункт приема»:

* Описание: локация, где можно сдать вторсырье.
* Атрибуты: адрес, время работы, ссылка на карту, имя пункта приема.
* Связан с Пользователями и Типами вторсырья (пользователи могу сдать различные виды вторсырья).

Описание объекта «Вид вторсырья»:

* Описание: виды вторсырья, которые могут быть сданы в пункты приема.
* Атрибуты: название вида вторсырья, начисляемые баллы за сдачу, количество новой продукции из 1 кг вторсырья, ссылка на изображение.
* Связана с Пользователями и Пунктами приема (пользователи сдают вторсырье в определенные пункты приема).

# 7 Порядок приемки и контроля

Процесс приемки и контроля приложения будет включать несколько ключевых этапов, которые позволят оценить соответствие программного обеспечения установленным требованиям и его готовность к эксплуатации.

Предварительное тестирование. До начала процедуры официальной приемки система должна пройти этап внутреннего тестирования. Это включает в себя:

* Модульное тестирование отдельных компонентов для выявления ошибок на ранних стадиях разработки.
* Интеграционное тестирование, чтобы убедиться, что все компоненты корректно взаимодействуют друг с другом.
* Функциональное тестирование, которое проверяет выполнение всех основных функций системы.

Альфа-тестирование. На данном этапе приложение предоставляется ограниченному кругу пользователей. Основное внимание уделяется проверке функциональности, производительности, безопасности и удобства использования.

Бета-тестирование. Цель бета-тестирования – выявить скрытые проблемы, связанные с использованием приложения в реальных условиях.

Приемочное тестирование. Это финальный этап, на котором проводится проверка приложения на соответствие техническому заданию и требованиям заказчика. В ходе приемочного тестирования выполняются следующие виды проверок:

* Функциональное тестирование: проверяется выполнение всех обязательных функций и отсутствие критических ошибок.
* Нагрузочное тестирование: проверка производительности системы при высокой нагрузке, чтобы убедиться в ее масштабируемости и способности.

Документационная проверка. В рамках процесса приемки проверяется наличие и корректность всей сопроводительной документации, включая руководство пользователя, техническую документацию, а также Swagger-документацию API.

Утверждение приемки. По результатам приемочного тестирования и документальной проверки составляется акт приемки, в котором фиксируются итоги всех тестов. Если приложение соответствует всем заявленным требованиям, он считается принятым и готовым к развертыванию и эксплуатации. В случае выявления критических недочетов, они фиксируются, и принимается решение о необходимости доработок.

Этапы контроля и приемки обеспечивают гарантию того, что приложение отвечает заявленным требованиям по функциональности, надежности и безопасности, а также готов к использованию конечными пользователями.

# Заключение

Приложение «EcoSort» – полностью функциональное приложения для предоставления пользователям возможности сдавать вторсырье и получать ключевую информацию о том, сколько новой продукции будет произведено из сданного материала, а также для ряда дополнительных функций.

За серверную часть отвечала платформа NodeJS, а именно фреймворк ExpressJS, который позволил реализовать быструю асинхронную серверную часть.

Клиентская часть была написана при помощи библиотеки ReactJS для языка JavaScript. Благодаря Redux ToolKit Query клиент могу делать асинхронные запросы к серверу, что в следствии привело к более быстрой загрузке всех данных на сайте. React позволил легко построить структуру клиентской части веб-приложения.

База данных MySQL позволила создать структуру, которая позволила эффективно хранить взаимосвязанные данные. База данных содержит функционал для работы с ней, также в ней описаны роли, которые ограничивают доступ к данным.

В итоге было разработано веб-приложение, которое не только предоставляет пользователям возможность сдавать вторсырье и получать ключевую информацию о том, сколько новой продукции будет произведено из сданного материала, но и предлагает ряд дополнительных функций. Пользователи могут легко ознакомиться с различными пунктами приема и видами вторсырья, которые они могут сдать. Они также имеют возможность обменивать накопленные баллы на доступные скидки и получать промокоды для их использования. Важным аспектом является возможность пользователями публиковать полезные статьи, редактировать их, а также просматривать статьи, опубликованные другими пользователями. Они могут оставлять комментарии к статьям, добавлять лайки к понравившимся материалам.

Тестирование проекта показало, что приложение функционирует правильно и все поставленные тесты прошло. В ходе тестирования ошибок не было найдено.

На этапе развертывания в сети Интернет были выполнены все необходимые настройки серверов и базы данных для обеспечения устойчивости и доступности приложения.

Конечный продукт предоставляет информацию о раздельном сборе бытовых отходов в домашних условиях и где их потом можно сдавать.

Веб-приложение является законченным программным продуктом, который реализует все перечисленные возможности в полной форме. Веб-приложение поддерживает архитектуру, которая позволяет в дальнейшем быстро и безопасно расширить функционал.