

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой № 82
д-р экон. наук, профессор

должность, уч. степень, звание

подпись, дата

Будагов А.С.

инициалы, фамилия

БАКАЛАВРСКАЯ РАБОТА

на тему Разработка информационной системы стартапа капшеринга (ВКР-С)

выполнена Архиповым Ростиславом Васильевичем
фамилия, имя, отчество студента в творительном падеже

по направлению подготовки 09.03.03 Прикладная информатика
код наименование направления

направленности 09.03.03
код наименование направленности

Прикладная информатика в экономике
наименование направленности

Студент группы № 8026 Р. В. Архипов
подпись, дата инициалы, фамилия

Руководитель
доцент, канд. техн. наук

должность, уч. степень, звание

подпись, дата

А. Д. Державина

инициалы, фамилия

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

УТВЕРЖДАЮ

Заведующий кафедрой № 82

д-р экон. наук, профессор

должность, уч. степень, звание

подпись, дата

А.С. Будагов

инициалы, фамилия

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ БАКАЛАВРСКОЙ РАБОТЫ

студенту группы

8026

номер

Архипову Ростиславу Васильевичу

фамилия, имя, отчество

на тему Разработка информационной системы стартапа капшеринга (ВКР-С)

утвержденную приказом ГУАП от

№

Цель работы: Разработка информационной системы стартапа капшеринга
на базе веб-технологий

Задачи, подлежащие решению: изучение лучших практик веб разработки и внедрения
информационных систем; анализ бизнес-процессов и разработка технических решений;
исследование рынка для выявления конкурентов и анализа их продукта и систем;
разработка сайта, содержащего информацию о стартапе и включающего в себя личный
кабинет; разработка базы данных;
оценка технико-экономической эффективности стартапа.

Содержание работы (основные разделы): введение, анализ предметной области,
проектирование информационной системы, разработка информационной системы
компании, финансовый план и оценка проекта, заключение, список использованных
источников, приложения.

Срок сдачи работы « 10 » июня 2024

Руководитель

доцент, канд. техн. наук

должность, уч. степень, звание

подпись, дата

А. Д. Державина

инициалы, фамилия

Задание принял(а) к исполнению

студент группы №

8026

подпись, дата

Р. В. Архипов

инициалы, фамилия

РЕФЕРАТ

Отчет 100 стр., 44 рис., 7 табл., 24 источников, 9 приложений.

Целью данной дипломной работы является разработка информационной системы стартапа капшеринга на базе веб-технологий

Объектом исследования в данной работе является процесс автоматизации бизнес-процессов стартапа капшеринга с использованием информационной системы.

Предметом исследования являются методы разработки информационных систем с помощью веб-технологий, интеграция их в стартап, а также системы учёта складских остатков.

Практическая значимость работы заключается в создании готового решения, которое может быть использовано малым и средним бизнесом сферы общественного питания.

В данной работе используются методы системного анализа, проектирования программных систем, программирования и интеграции различных сервисов.

Дипломная работа состоит из анализа предметной области, проектирования информационной системы, разработки информационной системы и финансового плана и оценки проекта. В анализе предметной области определяется целевая аудитория, рассматриваются существующие решения на рынке, ставится задача и проходит опрос потенциальных потребителей. В проектировании информационной системы происходит моделирование бизнес-процессов, связанных с информационной системы, выделяются требования к системе и описывается база данных. В разработке информационной системы описывается процесс разработки интерфейса и серверной части. Также в этой части был разработан минимально работоспособный продукт. В финансовом плане и оценке проекта произведены расчёты, обосновывающие выгоду использования разработанной информационной системы стартапа.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ	6
ВВЕДЕНИЕ	7
1 Анализ предметной области	9
1.1 Определение целевой аудитории.....	9
1.2 Конкурентный анализ	12
1.3 Постановка задачи.....	22
1.4 Анализ объемов рынка	23
1.5 Опрос потенциальных покупателей	25
2 Проектирование информационной системы	27
2.1 Моделирование бизнес-процессов	27
2.2 Требования к системе	31
2.2.1 Функциональные требования.....	31
2.2.2 Нефункциональные требования	31
2.3 База данных.....	32
2.4 Построение схемы базы данных.....	35
3 Разработка информационной системы	38
3.1 Разработка интерфейса	38
3.2 Разработка API.....	46
4 Финансовый план и оценка проекта	50
4.1 Финансовый план разработки ИС	50
4.2 Финансовый план проекта	53
ЗАКЛЮЧЕНИЕ	59
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	60
Приложения	64

Приложение А	64
Приложение Б	71
Приложение В.....	73
Приложение Г	74
Приложение Д.....	77
Приложение Е.....	80
Приложение Ж.....	90
Приложение З	93
Приложение И	96

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

Таблица 1 – Перечень сокращений

Сокращение	Расшифровка
API	Application Programming Interface (Программный интерфейс приложения)
JWT	JSON Web Token – Веб-токен формата JSON
REST	Representational State Transfer (Передача репрезентативного состояния)
СУБД	Система управления базами данных
MVP	Minimal Viable Product (Минимально жизнеспособный продукт)
ИС	Информационная система
УТП	Уникальное торговое предложение
БД	База данных
UI	User Interface (пользовательский интерфейс)
DTO	Data Transfer Object (объект передачи данных)
ФОТ	Фонд оплаты труда
NPV	Net Present Value (Чистая приведенная стоимость)

ВВЕДЕНИЕ

В современном мире вопросы экологии, устойчивого развития, переработки и более грамотного использования ресурсов всё чаще и больше вызывают интерес и внимание общественности. Также к решению проблем экологии активно присоединяются представители бизнеса, государств и некоммерческих организаций [1]. Избыточное производство и недостаточные объёмы переработки, в том числе за счёт одноразовых стаканчиков – одна из самых выраженных проблем [2]. Стартапы, предлагающие инновационные решения для снижения экологического следа, становятся все более популярными.

Сервис капшеринга – это специализированная коммерческая организация (стартап), осуществляющая услуги по аренде и последующему обмену многоразовых стаканов различных ёмкостей, сделанных из экологичного материала – первичного полипропилена. Данная услуга может производиться как в кафе, так и на мероприятиях любого типа. Немаловажно отметить, что главная задача стартапа как такового существенно снизить количество отходов от одноразовой посуды.

Актуальность проекта состоит в том, что на данный момент в схожих по роду деятельности компаниях отсутствует комплексное решение по реализации стаканов. Из-за этого аналогичные компании плохо масштабируемы, неклиентоориентированы и имеют слабый потенциал в развитии и популяризации экологических решений. Также следует отметить ряд факторов, которые являются ключевыми в обосновании актуальности стартапа – это экологическая значимость, социальная ответственность и поддержка государства.

В настоящее время Web2 технологии достаточно развиты и предоставляют возможность разрабатывать информационные системы, выполняющие свою первоочередную задачу, а именно хранить и обрабатывать большие объёмы данных и транзакции в режиме реального времени. Стоит

отметить, что технологии позволяют информационным системам обладать высокими показателями безопасности и способностью к масштабируемости.

Разработка информационной системы, которая позволит пользователям брать в аренду многоразовые стаканы для напитков, а бизнесу быстро и качественно осуществлять их оборот является основной целью проекта.

Важно отметить, что для успешной реализации цели проекта необходимо изучить и применить на практике современные веб-технологии и практики разработки и внедрения информационных систем, методы разработки NoSQL баз данных. Так же понадобится анализ бизнес-процессов для моделирования бизнес-логики и структуры стартапа. Для оценки экономической эффективности проекта потребуется изучить сведения о рынке, потенциале масштабируемости и мнении целевой аудитории.

Ключевая бизнес-задача проекта – реализация экономически выгодной, удобной и экологичной системы аренды многоразовых стаканов, которая сможет популяризовать разумное потребление в сфере общественного питания и обеспечить сокращение использования бумажных стаканов в кафе, кофейнях и других общественных местах.

Из всего вышеперечисленного можно сделать вывод о высокой актуальности и потенциале проекта разработки информационной системы стартапа капшеринга. Это не только способствует решению глобальных проблем, но и даёт конкурентное преимущество перед похожими стартапами, что в условиях растущего спроса на эко-решения открывает большие бизнес-возможности.

1 Анализ предметной области

На сегодняшний день в мире наблюдается высокий социальный интерес к проблемам экологии.

Исходя из данных различных исследований, ежегодно производится около 3 сотен миллионов тонн пластика по всему миру, основная часть которого приходится на одноразовую посуду. Переработка подобных изделий составляет лишь 20% от количества отходов, попадающих на полигоны [2]. Капшеринговая система предоставляет возможность отслеживания пластиковой посуды и более качественной обработки, включая и повторное использование.

На территории Европы социум хорошо осведомлён об экологических проблемах и тоже стремится их решить. Так в Германии в 2018 году появился капшеринговый сервис Resur, который сам производит стаканы и продаёт их в кафе. Затем они выдаются по залоговой системе 1 евро за стакан.

Сейчас всё большее количество стран вводят законы, направленные на регулирование количества пластиковых отходов и поощрение их переработки, что создаёт благоприятные условия для развития экологических проектов и инициатив [3].

В России действует федеральный закон "Об отходах производства и потребления" от 24.06.1998 N 89-ФЗ, который определяет правовые основы обращения с отходами производства и потребления в целях предотвращения вредного воздействия отходов производства и потребления на здоровье человека и окружающую среду [5].

1.1 Определение целевой аудитории

Для успешного запуска и развития стартапа, занимающегося капшерингом стаканов, необходимо тщательно определить целевую аудиторию. Понимание того, кто является потенциальными пользователями, позволит настроить стратегии маркетинга и продвижения продукта так, чтобы

они отвечали потребностям и ожиданиям клиентов. В контексте стартапа, который предлагает инновационное решение для сокращения использования одноразовой посуды через систему аренды многоразовых стаканов, целевую аудиторию можно разделить на несколько ключевых групп.

Экологически осведомленные потребители. Эта группа включает в себя людей, активно интересующихся экологическими вопросами и ищущих способы сократить свой экологический след. Они склонны выбирать продукты и услуги, которые способствуют устойчивому развитию. Возраст этой части целевой аудитории чаще всего от 25 до 35 лет. Эти люди активно ведут социальные сети, для обмена информацией и заинтересованы в поиске способов экологического потребления, часто ищут экологически чистые альтернативны и товары с меньшим воздействием на окружающую среду, а также участвуют в экологических инициативах.

Студенты и молодые профессионалы Эта группа живет в городах и районах с высокой степенью урбанизации, часто пользуется общественными пространствами типа кафе и коворкингов, где могут пить кофе или чай в течение дня. Возраст данной аудитории колеблется от 18 до 25 лет. Ведут активную социальную жизнь, имеют высокий уровень мобильности, открыты к нововведениям, часто используют мобильные приложения для управления своими ежедневными делами

Корпоративные клиенты – это представители малого бизнеса, стремящиеся улучшить свой общественный образ и внедрить устойчивые практики в свои бизнес-процессы. В основе своей это кафе, коворкинги, креативные кластеры, общественные пространства, офисы и учебные заведения. Представители этой части целевой аудитории заинтересованы в создании позитивного имиджа, могут поддерживать экологические решения исходя из соображений экономии и привлечения новой аудитории. Так же заинтересованы во введении новых технологий и практик.

Посетители крупных городских и культурных мероприятий Люди, посещающие фестивали, концерты, лекции, конференции и другие массовые

мероприятия, на которых традиционно используется много одноразовой посуды, также могут стать важной аудиторией для капшеринга стаканов. Данный кластер целевой аудитории представляют собой люди от 18 до 45 лет, которые ведут активное участие в культурной жизни города, проявляют интерес к музыке и искусству, бизнесу и другим общественным событиям. Обладают высокой социальной активностью и развитым чувством ответственности за сохранение чистоты окружающей среды.

Маркетинговые и продуктовые стратегии

Для каждой из этих групп следует разработать специализированные маркетинговые и коммуникационные стратегии, которые помогут найти общие ценности и точки контакта с целевой аудиторией, а также чтобы наиболее точно понять, как взаимодействовать с той или иной частью целевой аудитории следует расставить акценты на коммуникациях, ознакомиться с которыми можно в таблице 2.

Таблица 2 – Поведенческие и маркетинговые стратегии

Номер п/п	Название группы	Основные транслируемые компанией ценности	Точка контакта
1	Экологически осведомлённые потребители	Использование экологически чистых материалов, вклад каждого пользователя в защиту окружающей среды	Социальные сети, ивенты в кафе или на мероприятиях
2	Студенты и молодые профессионалы	Использование экологически чистых материалов, популяризация эко- инициатив,	Социальные сети, коллаборации с университетами и распространёнными сетями кафе,

		инновационные решения, гибкость и мобильность, удобство использования	предоставление скидок
3	Корпоративные клиенты	Использование экологически чистых материалов, культурное вовлечение, экономия на расходных материалах	Бизнес-мероприятия, бизнес-встречи, онлайн-предложения, социальные сети
4	Посетители крупных городских и культурных мероприятий	Использование экологически чистых материалов, культурное вовлечение,	Бизнес-мероприятия, коллаборации с организаторами культурно-массовых мероприятий

Определение целевой аудитории и адаптация к ней всех аспектов бизнеса, от продукта до коммуникаций, играет ключевую роль в успехе стартапа [4]. Эффективное взаимодействие с целевой аудиторией позволит не только достичь коммерческого успеха, но и способствовать решению глобальных экологических проблем, что в свою очередь может привести к улучшению качества жизни в городах и на планете в целом.

1.2 Конкурентный анализ

Понимание рынка и конкурентной среды является критически важным аспектом для любого стартапа, включая проект по каппшерингу стаканов. Прежде чем приступить к разработке MVP и стратегии вывода продукта на рынок, необходимо провести тщательный анализ конкурентов. Это поможет не только выявить потенциальные угрозы и возможности, но и определить уникальные предложения, которые могут выделить стартап среди других игроков в данной нише.

Сервисов, имеющих информационную систему, которая позволяет регистрироваться и аутентифицироваться в системе, имеет механику взятия и сдачи арендованного стакана потребителем, а также для бизнес-пользователей заказывать поставку и отгрузку, на рынке критически мало, рассмотрим некоторые из них.

Recup – крупнейший сервис капшеринга на рынке Европы, но не распространённый на территории России. Не имеет информационной системы и предлагает оставлять залог с функцией возврата залога при возврате тары, однако имеет хороший брендинг и маркетинговую стратегию. Широко распространена по всему рынку Европы и представляет хорошего кандидата на сравнение с реализуемым стартапом. На рисунках 1-2 представлены переведённые на русский язык скриншоты с сайта компании, объясняющие систему стартапа.

Для потребителей: Простота в использовании.

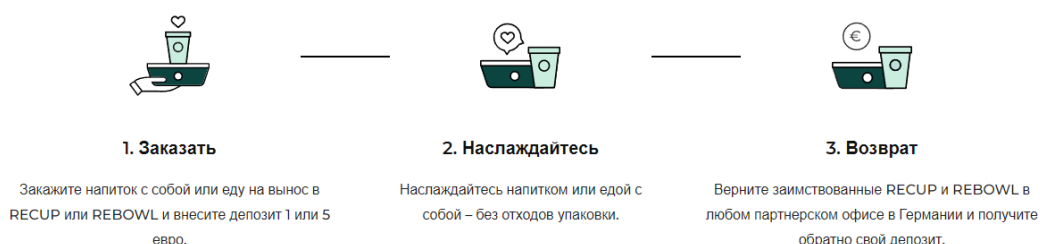


Рисунок 1 – Система стартапа Recup для потребителей

Для ресторанов: почему я должен предлагать RECUP и REBOWL?



Рисунок 2 – Система стартапа Rescup для ресторанов

Нарру Сип – отечественный стартап, концентрирующийся на работе с фестивалями. Использует аналогичную систему выдачи-возврата посуды по сравнению с предыдущим конкурентом. На рисунке 3 представлена информация с сайта компании.

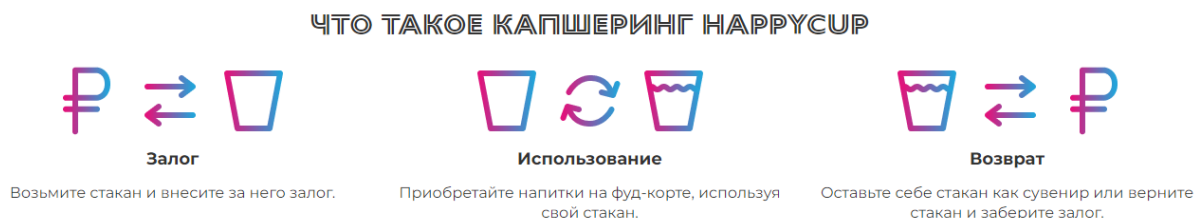


Рисунок 3 – Система стартапа Нарру Сип

Для связи с менеджерами проекта предлагается заполнить обратную форму и ждать ответ. На рисунке 4 можно ознакомиться с данной формой.

ФОРМА ДЛЯ СВЯЗИ
ЕСЛИ У ВАС ЕСТЬ ВОПРОСЫ, ОСТАВЬТЕ СВОИ КОНТАКТЫ И МЫ СВЯЖЕМСЯ С ВАМИ

<p>Ваше имя *</p> <input type="text"/>	<p>Сообщение</p> <input type="text"/>
<p>Телефон *</p> <input type="text"/>	
<p>E-Mail для связи *</p> <input type="text"/>	

☒ Я подтверждаю своё согласие на [обработку персональных данных](#) и ознакомился с [политикой конфиденциальности](#).

Отправить

Рисунок 4 – Форма обратной связи Нарру Сип

ВЕРНИ СТАКАН – работающий Екатеринбургский стартап эколо-активистов, однако не имеющий информационной системы. На лендинге проекта представлена карта проекта, ссылки на группу ВКонтакте. Ссылка «Стать партнером» ведет на мессенджер WhatsApp. На рисунке 5 изображен футер с контактными лицами стартапа. На рисунке 6 представлен скриншот из группы ВКонтакте.

БУДЕМ РАДЫ С ВАМИ РАБОТАТЬ

Написать WA

+7 993 520 4808

Позвонить

+7 912 651 4908

ВЕРНИ СТАКАН

Рисунок 5 – Футер лендинга с информацией

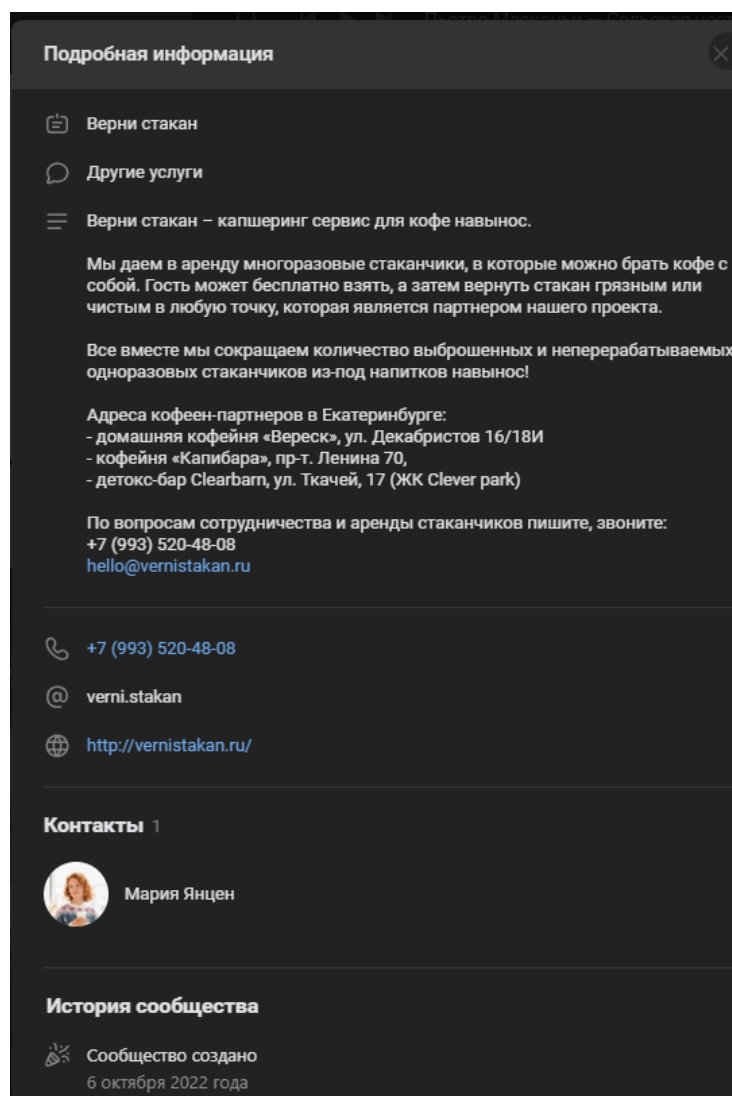


Рисунок 6 – Раздел информации в группе проекта VK

Обратка – запущенный Московский стартап, реализация ИС которого произведена посредством Telegram-бота, функции которой представлены на рисунках 7-8. Монетизация этого проекта реализована залоговой системой.

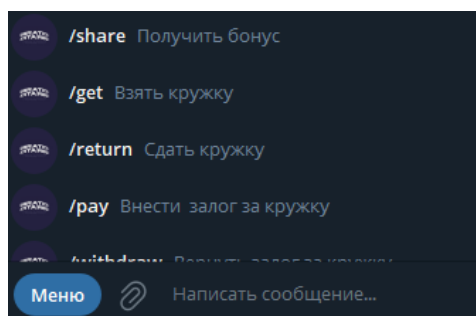


Рисунок 7 – Функционал TG-бота сервиса «Обратка»

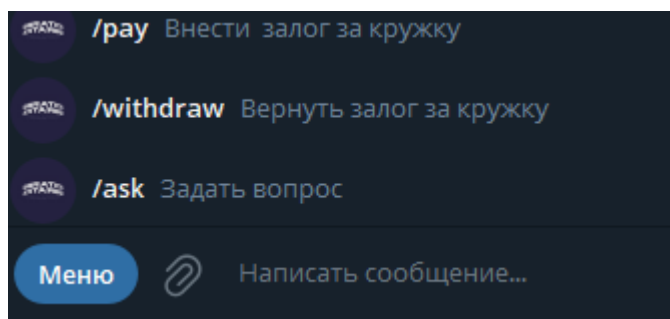


Рисунок 8 – Функционал TG-бота сервиса «Обратка»

CUPS – крупнейший на Российском рынке стартап, распространенный нескольких крупных городах. Имеет прототип информационной системы представлена на рисунках 9-11. Монетизация осуществлена залоговой системой.

КАК РАБОТАЕТ CUPS

Воспользоваться сервисом очень просто. Надо:

1. Войти в приложение CUPS (приложение доступно для Android) или на сайт **Cupshare.ru**, зарегистрироваться по номеру телефона (регистрация возможна, но не обязательна)
2. Показать бариста QR-код или назвать свой номер телефона
3. Заплатить 100 рублей и получить абонемент на аренду 10 стаканчиков
4. Бариста наливает вам напиток в многоразовый стаканчик CUPS со скидкой минимум 20 рублей
5. После использования стаканчика его можно сдать в любую кофейню участвующую в проекте
6. Когда абонемент закончится, можно повторить все шаги снова.

В приложении видно, сколько возможностей воспользоваться капшерингом у вас осталось и где находится ближайшая кофейня-партнер.

Рисунок 9 – Информация о сервисе CUPS

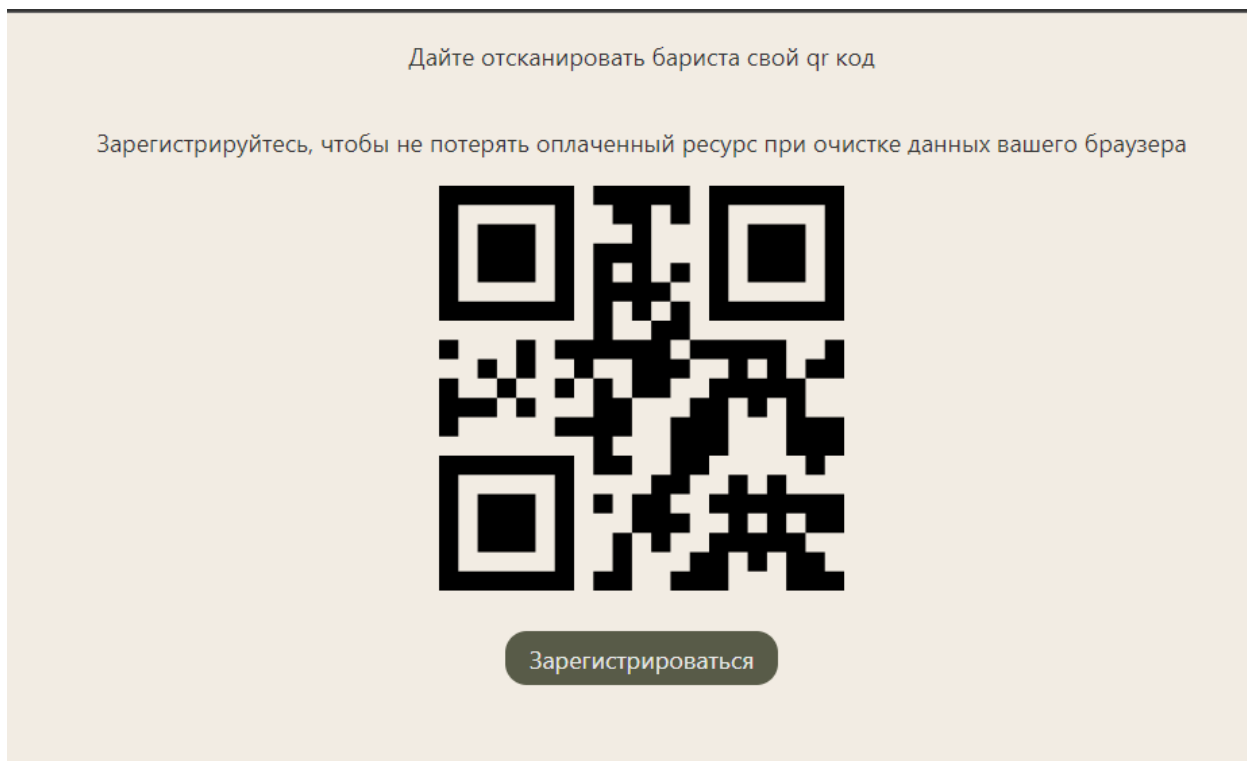


Рисунок 10 – Раздел «ID» сайта-ИС компании CUPS

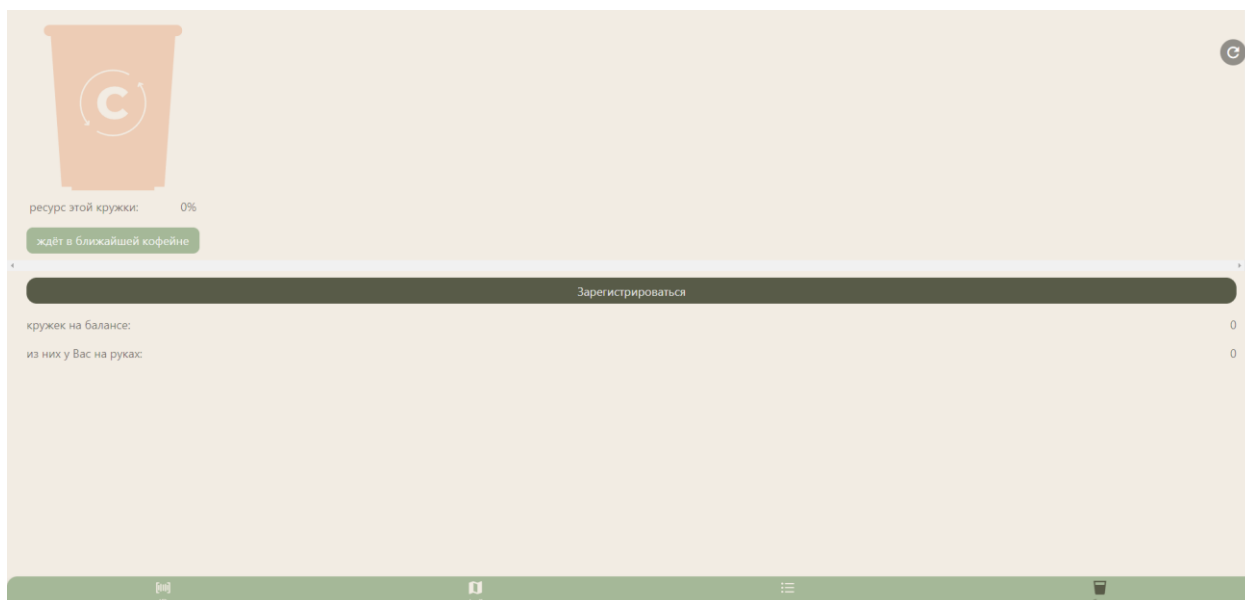


Рисунок 11 – Личный кабинет сервиса CUPS

Определим критерии, важные для потенциальных покупателей, для проведения анализа конкурентов:

1 Наличие информационной системы

Это ключевой фактор, которые представляет возможность для авторизации пользователей и их идентификации, автоматизирует процессы аренды стаканов и их поставки или отгрузки в кафе, а также реализует учёт возвращаемости стаканов.

2 Удобство пользования

Этот фактор необходимо оценивать, так как интуитивно понятные интерфейсы задают положительный тренд к узнаваемости бренда и формируют положительный пользовательский опыт.

3 Маркетинг и брендинг

Маркетинг и брендинг – залог успеха любой компании, так как это основные инструменты, привлекающие первых пользователей, формирующие у них лояльность бренду, а также привлекают потенциальных партнёров. Важно использовать креативные идеи и соответствовать трендам для обособления своих преимуществ перед конкурентами.

4 Экономическая выгода для бизнес-клиентов

Предложение экономически выгодных условий для предприятий малого бизнеса сферы общественного питания в данном случае основной способ взаимодействия с конечным потребителем, этот факт обуславливает необходимость создать выгодные условия сотрудничества с кафе.

5 Наличие системы подписок

Использование модели подписок позволяет больше стимулировать пользователей на покупки из-за относительно низкой цены, а также минимизирует затраты конечных потребителей и уменьшает количество процессов, выполняемых при залоговой системе [6].

6 Распространенность

Расширение географического покрытия данной бизнес-модели создает лучшие условия для масштабирования и благоприятно способствует на узнаваемость бренда.

Итоговый балл (S) программы будет рассчитываться формулой (1) как сумма произведений веса параметра (V_i) на степень его реализации (N_i):

$$S = \sum V_i * N_i \quad (1)$$

Степень реализации может варьироваться от 0 до 5, где 0 – не реализован полностью и 5 – реализован отлично.

Вес параметра также может варьироваться от 0 до 2, где 0 – параметром можно пренебречь и 2 – параметр критически важен для пользователя.

Каждый из вышеперечисленных сервисов обладает своими достоинствами и недостатками, многие из которых имеют критическое влияние на работоспособность сервиса. Представленные в таблице 3 данные наглядно отображают этот факт.

Таблица 3 – Сравнение похожих по роду деятельности стартапов

Параметр	Ресур	Нарру Сур	ВЕРНИ СТАКАН	Обратка	CUPS	Вес
Наличие ИС	0	0	0	3	3	2
Удобство пользования	5	2	2	3	3	2
Маркетинг и брендинг	5	2	3	3	4	1
Экономическая выгода для бизнес- клиентов	4	2	2	3	3	2
Наличие системы подписок	2	1	1	3	2	2
Распространенность	5	2	2	2	3	1
Итог	36	14	15	29	29	

Основывая на данных составленной таблицы, для большей наглядности были отрисованы графики (рисунок 12-13).

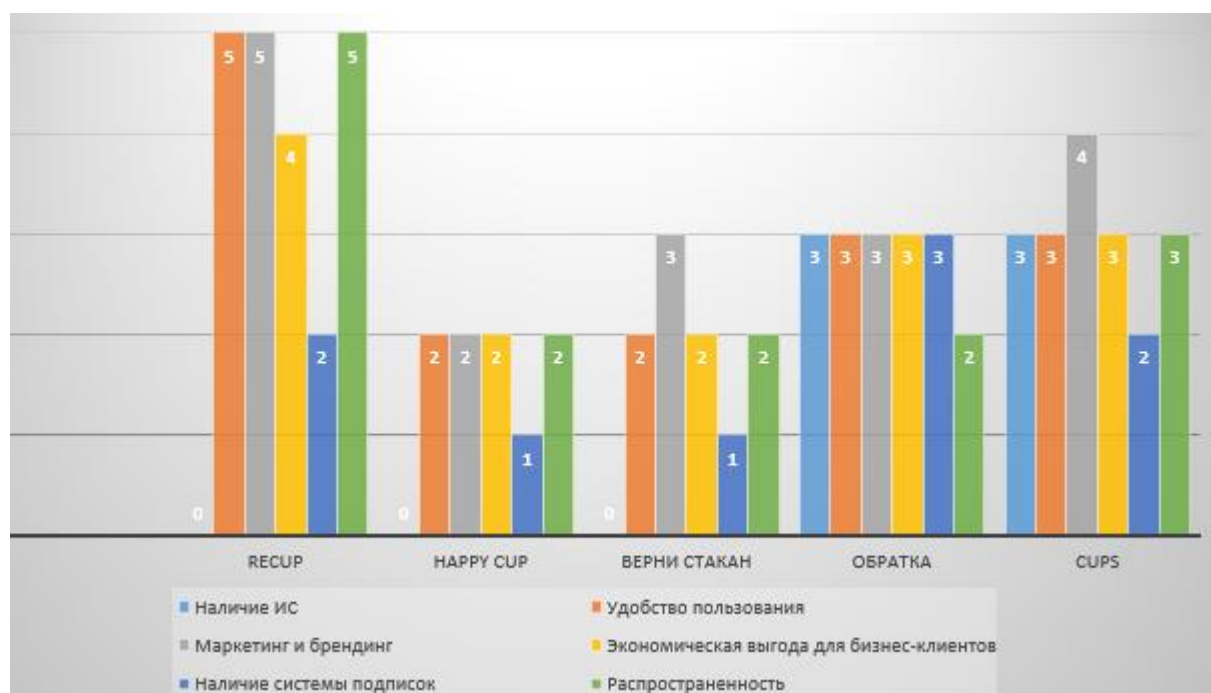


Рисунок 12 – Наглядное сравнение конкурентов по критериям



Рисунок 13 – Итоговые баллы

Из данных очевидно, что на рынке представленных решений существуют проблемы с развитой инфраструктурой предприятий, наличием ИС удобной в использовании, исходя из этого необходимо разработать собственную ИС, так как для Российского рынка не существует работоспособной системы, которая предоставляет систему подписок для

аренды многоразовых стаканов и обладает всеми необходимыми качествами для пользования ей как для конечного потребителя, так и для бизнес-клиентов.

Сравнительный анализ конкурентов предоставляет возможность понять на какие аспекты следует сделать упор при создании собственного проекта. Таким образом при запуске нужно сфокусировать внимание на повышении удобства использования, активном маркетинге, внедрении инновационных решений в технологическую основу проекта и активно расширять географический охват. Учет этих аспектов позволит не только выделиться среди конкурентов, но и создать стойкую основу для долгосрочного успеха и роста стартапа.

1.3 Постановка задачи

Стартап ShareCUP предлагает инновационный подход к разумному потреблению, с помощью многоразового использования стаканов в кафе и других общественных местах. Основная цель разработки ShareCUP – создание удобной, экологически чистой альтернативы одноразовым стаканам, которая помогает снижать количество отходов и улучшать экологическую обстановку в городах.

ShareCUP предлагает уникальную систему капшеринга стаканов, которая включает в себя не только аренду и возврат многоразовых стаканов, но и полноценную интеграцию с уникальной информационной системой, позволяющей пользователям легко находить ближайшие пункты выдачи или возврата стаканов через сайт, а также удобно брать и отдавать стаканы при помощи штрихкода в своём личном кабинете. Для корпоративных клиентов предлагается весь необходимый функционал, включающий в себя личный кабинет бизнес-пользователя и позволяющий заказывать и отгружать новые партии стаканов и удобно производить выдачу стаканов клиентам. Это подкрепляется экономией на расходных материалах, так как планируется сделать закупку многоразовых стаканов дешевле, чем одноразовых.

Цель разработки ShareCUP – не просто создать альтернативу существующим решениям по сокращению отходов, а предложить комплексный подход, который будет способствовать формированию новой культуры потребления и уменьшению экологического отпечатка на уровне всего общества. Это включает в себя образовательные кампании, сотрудничество с экологическими организациями, бизнес-форумами, сетями кафе и активное участие в экологических проектах.

В отличие от конкурентов, ShareCUP акцентирует внимание на социальной ответственности и технологичности процессов, связанных не только бизнес-логикой, но и производством. Это позволяет пользователям не только участвовать в программе, но и быть уверенными в том, что их выбор действительно экологичен и несёт пользу для окружающей среды.

1.4 Анализ объемов рынка

Анализ рынка по методике TAM, SAM и SOM (Total Addressable Market, Serviceable Available Market, Serviceable Obtainable Market) – это важный этап в оценке бизнес-потенциала стартапа ShareCUP. В этом анализе TAM определяет общий объем рынка для продукта или услуги, SAM указывает на часть TAM, которую компания может обслуживать, а SOM представляет сегмент SAM, который компания реально может завоевать в краткосрочной перспективе.

1. Total Addressable Market (TAM)

Для определения TAM необходимо оценить общее количество потребителей или объем продаж, который мог бы быть достигнут, если бы услуга была доступна для всех потенциальных потребителей во всех географических регионах.

Рынок: Все кафе, кофейни, а также конечные потребители в стране.

По данным TinkoffJournal количество кафе в России по состоянию на 2021 г. равно 91000, а количество человек, употребляющих кофе хотя бы раз в неделю в заведениях общественного питания около 35 миллионов человек [7].

В среднем в день расходуется около 300 стаканов в среднем по объемам продаж кафе. Умножив это количество на количество дней в году и на количество кафе, получим число использованных в год одноразовых стаканов, равное 10 миллиардам. Тогда TAM для ShareCUP составляет 10 миллиардов использований стаканов в год.

2. Serviceable Available Market (SAM)

SAM — это часть TAM, которую компания может обслужить, исходя из своих текущих возможностей и рыночной стратегии.

ShareCUP на первых стадиях разработки географически ограничен и сосредоточен на крупных городах с высоким потреблением напитков на вынос, где более высокая концентрация целевой аудитории и доступность инфраструктуры для капшеринга.

Если 30% от TAM находятся в крупных городах, то SAM для ShareCUP составляет 3 миллиарда использований стаканов в год.

3. Serviceable Obtainable Market (SOM)

SOM — это реальный объем рынка, который ShareCUP планирует завоевать, учитывая конкуренцию и ограничения в начальной стадии развития.

Планируется охватить 5% от SAM в первые несколько лет после запуска из-за конкуренции и постепенного наращивания клиентской базы.

Таким образом, SOM для ShareCUP составляет 150 миллионов использований стаканов в год.

Этот анализ позволяет понять потенциальный размер рынка для стартапа и определить, на какой объем продаж можно реально рассчитывать в ближайшие годы. Эти данные помогут в планировании производственных мощностей, маркетинговой стратегии, расчёте финансового плана и привлечении инвестиций.

1.5 Опрос потенциальных покупателей

Для проведения проблемных интервью были составлены гипотезы, которые можно будет подтвердить или опровергнуть по результатам опроса:

1 «Большинство опрашиваемых среди целевой аудитории предпочтут переплатить, чтобы быть уверенными в экологичности».

2 «Введение дифференцированных тарифов подписки, включая скидки для студентов и молодежи увеличит количество заинтересованных в стартапе»

3 «Партнёрство с популярными кафе для предложения многоразовых стаканов привлечет внимание более старшей возрастной категории»

Далее был разработан перечень вопросов, которые позже были представлены перед респондентами. Подробные результаты анкетирования представлены в приложении А.

В опросе приняли участие 158 человек в возрасте от 18 до 36 лет. Оценивались такие пункты как частота покупки напитков в одноразовых стаканах, частота посещения кафе, обеспокоенность количеством отходов от одноразовой посуды, осведомлённость о вреде пластиковых отходов, важность использования экологически чистых продуктов и услуг, готовность респондентов к переплатам за экологически чистые решения, преимущества и недостатки использования многоразовых стаканов, предпочтение клиентов к использованию многоразовых стаканов, готовность внести залог, готовность вернуть стакан, убеждение к использованию, важность географической распространенности, мнение о перспективе популярности данного решения, факторы популяризации, корреляция использования капшеринга от места, дополнительные удобства и зависимость самочувствия клиента от осведомленности о экологической пользе.

Исходя из результатов этого опроса, можно наблюдать картину возрастного разброса опрошенных респондентов, большинство из них это

студенты возраста 18-22 года, вторая по численности группа – студенты 22-26 лет, 4,4% опрошенных люди старше 30 лет.

Большинство опрошенных редко покупают напитки в одноразовой таре, однако около 37% делают это несколько раз в неделю. 74,1% опрошенных посещают кафе или кофейни несколько раз в неделю. 76% респондентов обеспокоены количеством пластиковых отходов от одноразовой посуды и хорошо осведомлены о их вреде для окружающей среды. Абсолютное большинство считает важным использование экологических решений и готовы за это переплачивать.

Примерно 77% опрошенных видят в использовании многоразового стакана преимущества в виде снижения экологического воздействия и удобства использования. А недостатками – нежелание носить стакан с собой и неудобство мытья. Тем не менее абсолютное большинство респондентов предпочли бы использовать многоразовый стакан, если бы это было удобно и доступно и чувствовали бы себя при этом участниками защиты окружающей среды.

Большинство респондентов считает, что использование многоразовых стаканов может стать популярным среди большинства людей при условии повышения осведомленности и улучшения доступности, а также введения автоматизированной системы и скидок при ее использовании.

Вышеперечисленные выводы можно обобщить. Большинство молодых людей осведомлены о проблемах окружающей среды, в частности, от одноразовой тары и могли бы стать потенциальными клиентами сервиса капшеринга ShareCUP так как хотят нести своими решениями пользу окружающей среде, могут позволить себе переплатить за экологически чистое решение и думают, что данное решение может стать популярным.

2 Проектирование информационной системы

2.1 Моделирование бизнес-процессов

Для тщательного анализа происходящих в системе бизнес-процессов на этапе проектирования была разработана концептуальная модель, которая впоследствии была декомпозирована на уровень функции и задач.

На уровне системы предполагается целевое событие – Доступ к пользованию системой капшеринга. На рисунке 14 представлена концептуальная модель ИС на уровне системы в нотации IDEF0.

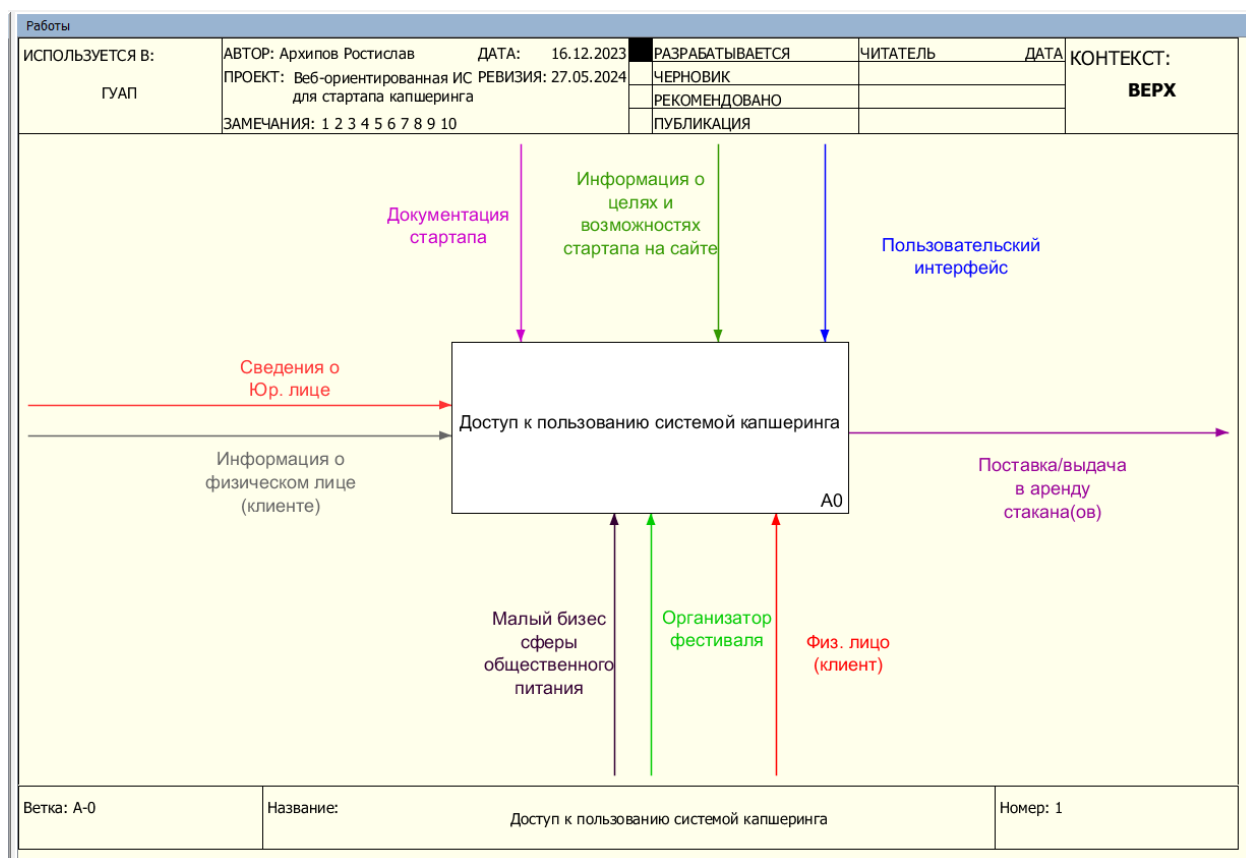


Рисунок 14 – Концептуальная модель ИС на уровне системы

Для моделирования уровня функций, нужно разбить целевую функцию уровня системы на подпроцессы, результат этого действия представлен на рисунке 15.

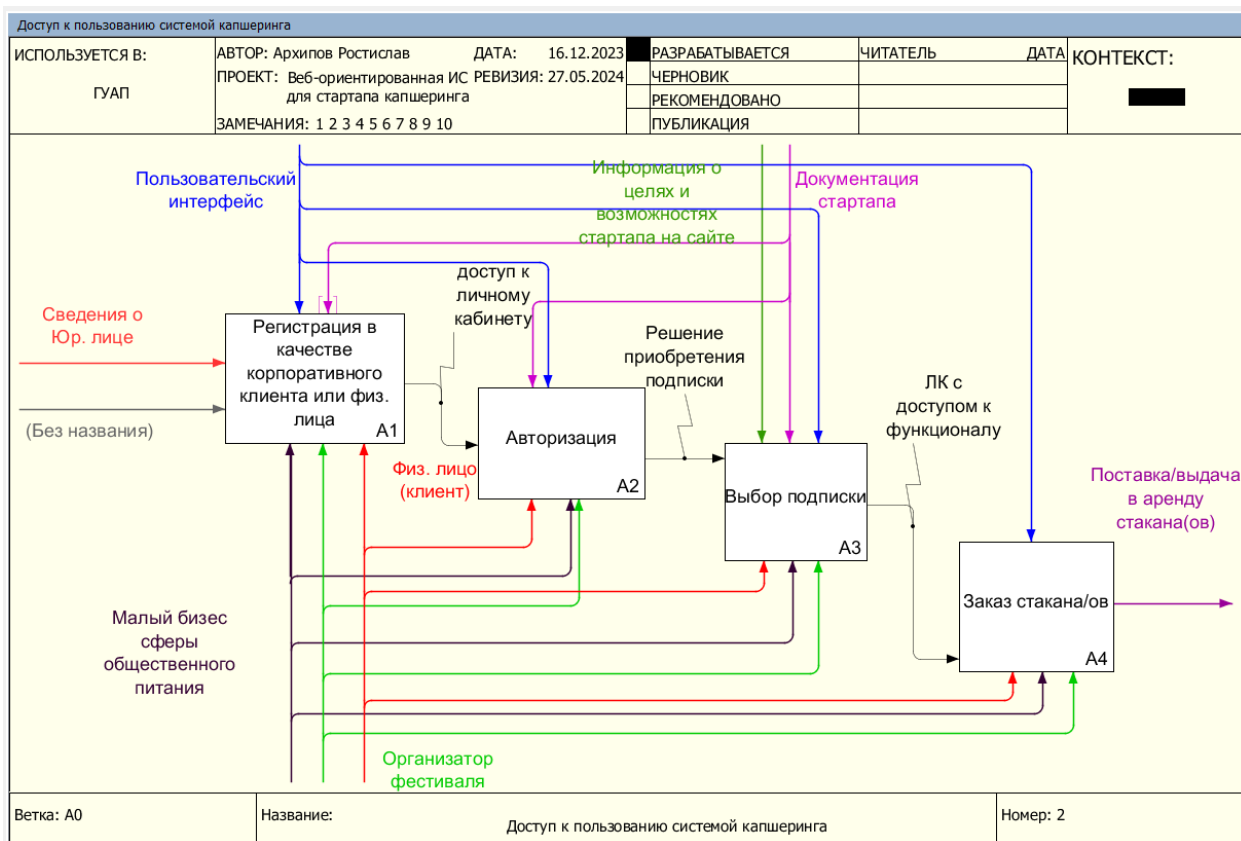


Рисунок 15 – Концептуальная модель ИС на уровне функций

На рисунках 16-19 представлена декомпозиция каждой функции на уровень задач. Этот уровень завершает моделирование бизнес-процессов в нотации IDEF0 для информационной системы капшеринга ShareCUP.

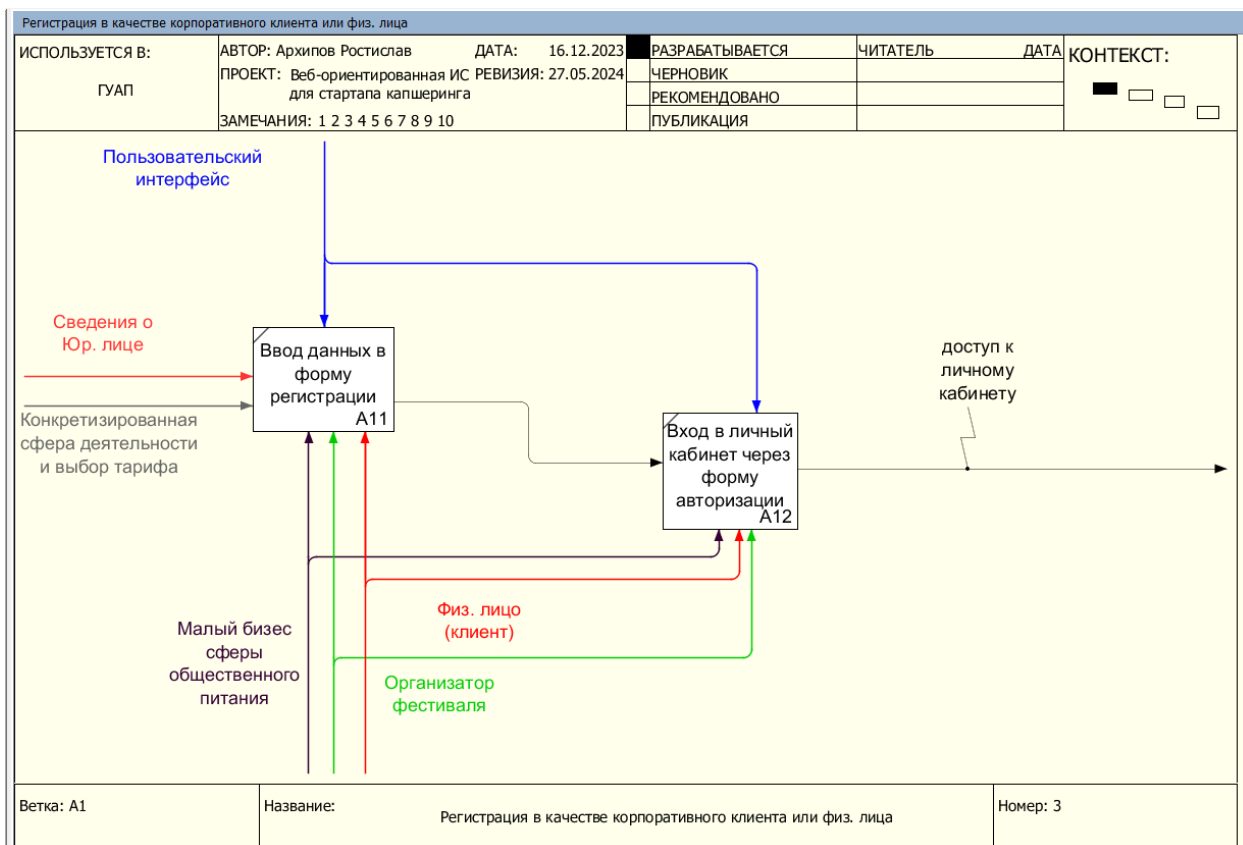


Рисунок 16 – Декомпозиция функции «Регистрация» на уровень задач

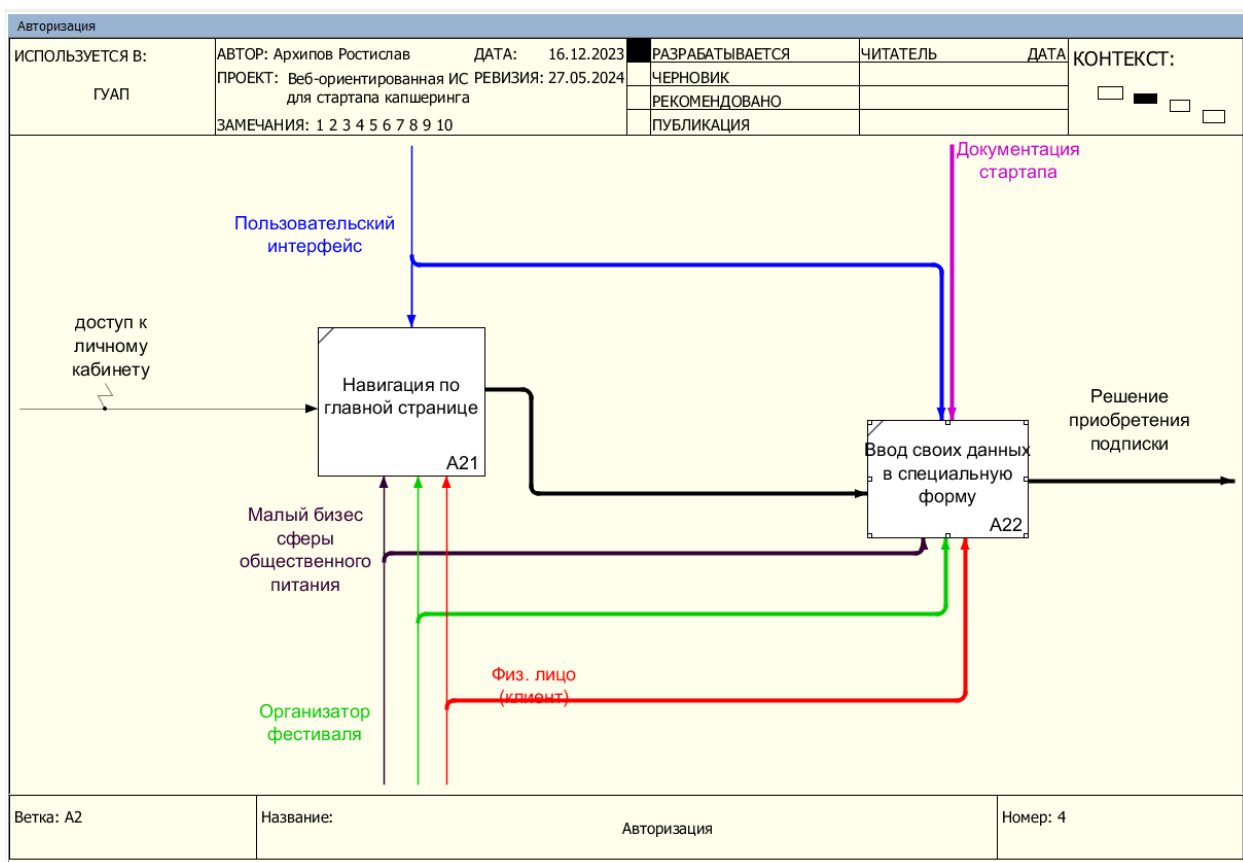


Рисунок 17 – Декомпозиция функции «Авторизация» на уровень задач

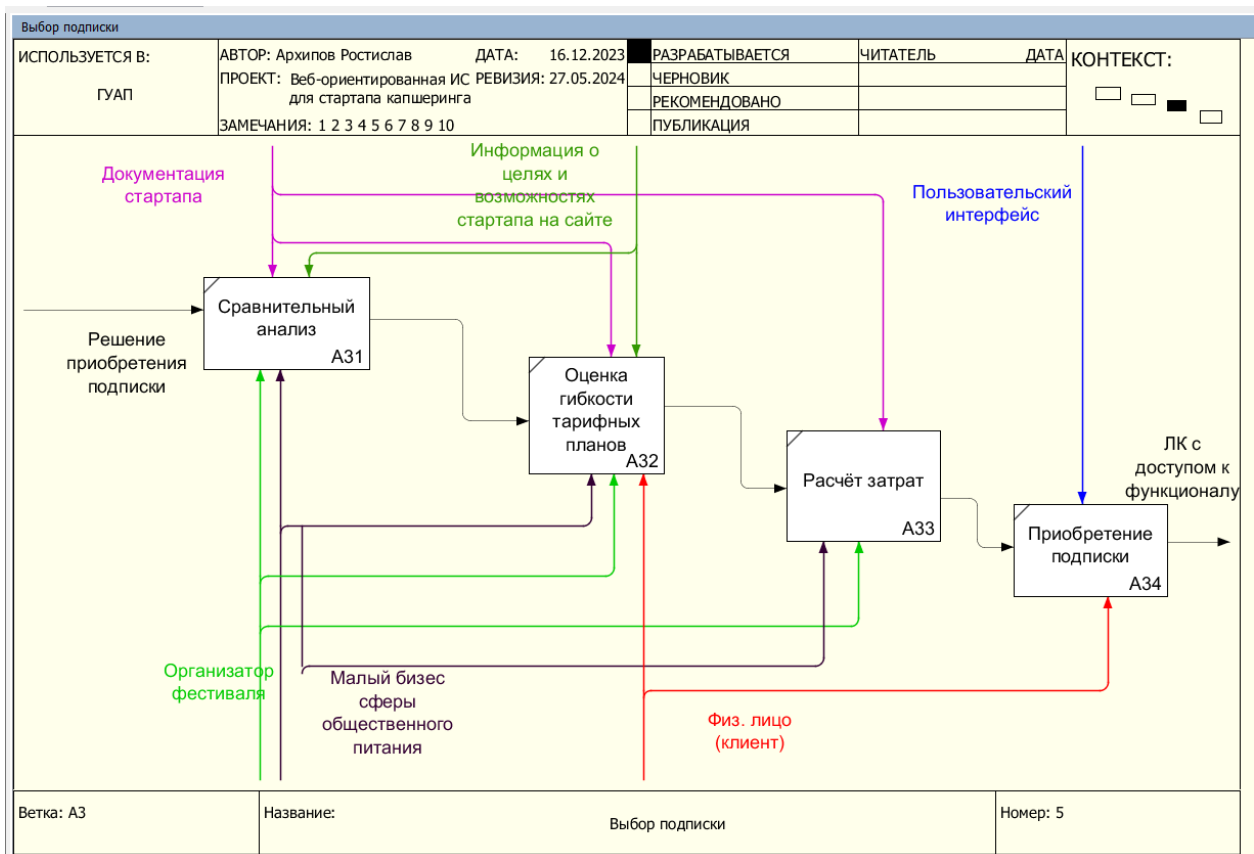


Рисунок 18 – Декомпозиция функции «Выбор подписки» на уровень задач

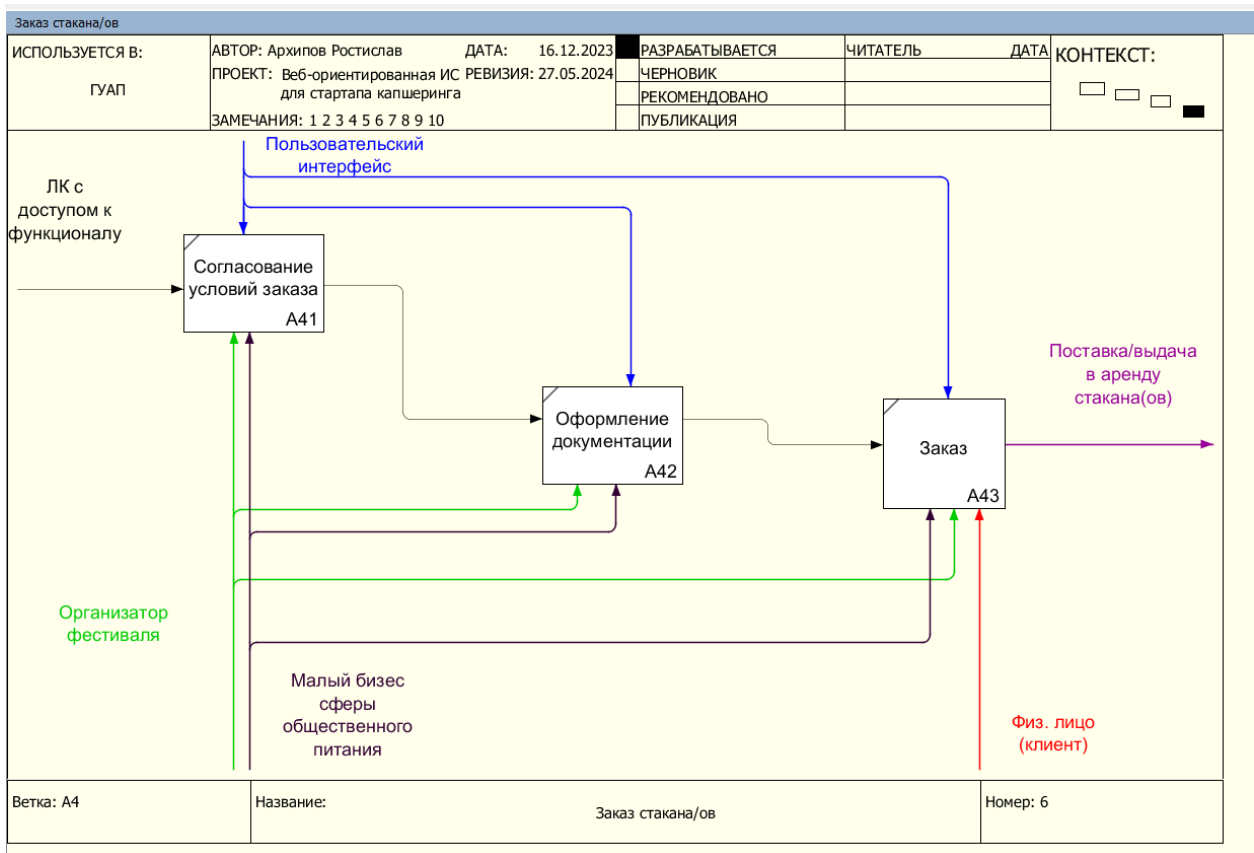


Рисунок 19 – Декомпозиция функции «Заказ стаканов» на уровень задач

Таким образом получаем бизнес-логику информационной системы, что значительно облегчит процесс дальнейшего проектирования и разработки.

2.2 Требования к системе

Для наиболее четкого достижения результатов разработки были проанализированы и сформулированы требования для проектируемой архитектуры информационной системы. Функциональные требования описывают сценарии взаимодействия пользователя и системы. Нефункциональные в свою очередь – описывают ограничения, которые должны соблюдаться, стоит отметить, что они не относятся к поведению системы в целом.

2.2.1 Функциональные требования

Целью данного дипломного проекта является разработка информационной системы стартапа капшеринга, которое удовлетворит все ключевые функциональные возможности выявленные в процессе исследования предметной области. Таким образом задачу можно свести к разработке следующих функциональных требований:

- 1 Возможность регистрации и авторизации пользователей.
- 2 Возможность изменять данные аккаунта.
- 3 Возможность поставки и отгрузки для бизнес-пользователя.
- 4 Возможность выбрать подписку.
- 5 Возможность сгенерировать штрих код конечным пользователем.
- 6 Возможность записать стакан на конкретного человека по штрих коду.

2.2.2 Нефункциональные требования

На основании указанных задач, можно выделить требования к системе в целом:

- 1 Удобство использования. UI должен быть унифицированным и понятным в использовании.

2 Разграничение доступа к системе. В системе должны быть предусмотрены несколько типов аккаунтов пользователей, в зависимости от которых будет доступен тот или иной функционал системы.

3 Шифрование данных. При взаимодействии клиентской части приложения с серверной должен использоваться расширенный протокол передачи данных с использованием шифрования HTTPS.

4 Надёжность. Приложение может и должно продолжить работу в случае возникновения ошибки.

5 Открытый программный код. Исходный код приложения должен быть размещён на открытом репозитории для исключения присутствия уязвимостей системы и вредоносных компонентов.

Для реализации всех вышеперечисленных требований и бизнес-процессов необходимо разделить информационную систему на две части: веб-приложение и API-сервис. API-сервис (бэкенд) представляет собой серверное приложение, предоставляющее программный интерфейс в архитектурном стиле REST, а веб-приложение (фронтенд) сайт с поддержкой адаптивности.

2.3 База данных

Для хранения и взаимодействия с информацией была выбрана документно-ориентированная база данных MongoDB. Её главная цель – хранение иерархических структур данных, реализуется при помощи NoSQL технологии. В фундаменте баз данных такого типа лежат хранилища документов, имеющих структуру дерева [8]. Структура начинается с корневого элемента и содержит в себе несколько узлов, символизирующих листья иерархии. Листовые узлы содержат в себе информацию, которая при добавлении новой индексируется, это позволяет всегда иметь быстрый доступ к данным даже в сложной структуре. Интерфейс программирования приложений MongoDB дает возможность выполнять поиск документов и их частей, используя ключи или их значения. В отличие от реляционных баз

данных, запросы к документным хранилищам могут возвращать части множества документов, не требуя их полной загрузки в оперативную память [9].

Главные особенности MongoDB:

1 Данные хранятся в виде коллекций и документов, поэтому в такой БД не требуется описания схемы таблиц и отношений между ними, как в реляционных БД.

2 Это кроссплатформенная, документно-ориентированная база данных, использующая подход NoSQL, а также имеющая открытый исходный код.

3 Данные хранятся в формате бинарной формы представления JSON- файлов – BSON.

4 Зачастую соединение производится при сохранении данных путем объединения документов, поэтому между коллекциями нет сложных соединений.

5 У коллекций не обязательно должна быть схожая структура. У одного документа может быть один набор полей, в то время как у другого документа – совершенно другой (как тип, так и количество полей).

В одном документе могут быть поля разных типов данных, которые не нужно приводить к одному. Основное преимущество MongoDB заключается в том, что она может хранить любые данные, но эти данные должны быть в формате JSON [10].

Для поддержания единого языка в рамках разрабатываемого сервиса и более глубокого понимания предметной области, выделим следующие сущности, с которыми придётся работать в дальнейшем в рамках разрабатываемой информационной системы стартапа капшеринга.

Пользователь – это основная сущность, содержащая информацию о пользователях системы.

Пользователь – это клиент сервиса капшеринга, который непосредственно взаимодействует с пользовательским интерфейсом и может

являться как физическим, так и юридическим лицом. Он обладает следующими характеристиками:

Атрибуты:

- email: string (required, unique) — уникальный электронный адрес пользователя.
- password: string (required) — зашифрованный пароль пользователя.
- firstName: string (required) — имя пользователя.
- lastName: string (required) — фамилия пользователя.
- company: boolean (default: false) — флаг, указывающий, является ли пользователь представителем компании.
- companyInn: string — ИНН компании, если пользователь представляет компанию.
- companyOgrn: string — ОГРН компании.
- companyKpp: string — КПП компании.
- companyAddress: string — адрес компании.
- companyName: string — название компании.
- createdAt: Date (default: Date.now()) — дата и время регистрации пользователя.
- isAdmin: boolean (default: false) — определяет, обладает ли пользователь административными правами.

Подписка - Сущность, отражающая информацию о подписке пользователя на услуги капшеринга.

Атрибуты:

- type: enum ['year', 'month'] — период подписки (годовая или месячная).
- userId: Types.ObjectId (ref: 'User') — ссылка на пользователя, который владеет подпиской.
- cupsAvailable: number — общее количество стаканов, доступное пользователю по подписке.
- cupsCurrent: number (default: 0) — текущее количество стаканов, доступных пользователю.

Уникальный ключ - Сущность для хранения приватных ключей, которые могут использоваться для аутентификации или других целей безопасности.

Атрибуты:

- user: string — идентификатор пользователя, к которому привязан ключ.
- key: string (unique) — уникальный приватный ключ.
- createdAt: Date (default: Date.now()) — дата и время создания ключа.

Стаканы - это сущность, представляющая данные о транзакциях с использованием стаканов в системе капшеринга.

Атрибуты:

- user: string — идентификатор пользователя, который осуществляет операцию со стаканами.
- amount: string — количество стаканов, затронутых в операции.
- type: enum ['order', 'return'] — тип операции: заказ стаканов или их возврат.
- approve: boolean (default: false) — статус подтверждения операции администратором или системой.

2.4 Построение схемы базы данных

Описание базы данных:

1. Таблица Cups

- user: string — пользователь.
- amount: string — количество.
- type: string (enum: ['order', 'return']) — тип действия (заказ или возврат).
- approve: boolean (default: false) — одобрение.

2. Таблица Subscription

- type: string (enum: ['year', 'month']) — тип подписки (годовая или месячная).
- userId: Types.ObjectId (ref: 'User') — идентификатор пользователя, ссылка на таблицу User.

- cupsAvailable: number — доступное количество стаканов.
- cupsCurrent: number (default: 0) — текущее количество стаканов.

3. Таблица User

- email: string (required, unique) — электронная почта пользователя.
- password: string (required) — пароль пользователя.
- firstName: string (required) — имя пользователя.
- lastName: string (required) — фамилия пользователя.
- company: boolean (default: false) — флаг, указывающий, является

ли пользователь компанией.

- companyInn: string (default: null) — ИНН компании.
- companyOgrn: string (default: null) — ОГРН компании.
- companyKpp: string (default: null) — КПП компании.
- companyAddress: string (default: null) — адрес компании.
- companyName: string (default: null) — название компании.
- createdAt: Date (default: Date.now()) — дата создания пользователя.
- isAdmin: boolean (default: false) — флаг, указывающий, является ли

пользователь администратором.

4. Таблица PrivateKey

- user: string — пользователь.
- key: string (required, unique) — уникальный ключ.
- createdAt: Date (default: Date.now()) — дата создания.

Связи между таблицами

– Subscription.userId ссылается на User._id, что обозначает связь между подписками и пользователями.

– Cups.user ссылается на User.email или идентификатор пользователя, что обозначает связь между стаканами и пользователями.

– PrivateKey.user также может ссылаться на User.email или идентификатор пользователя.

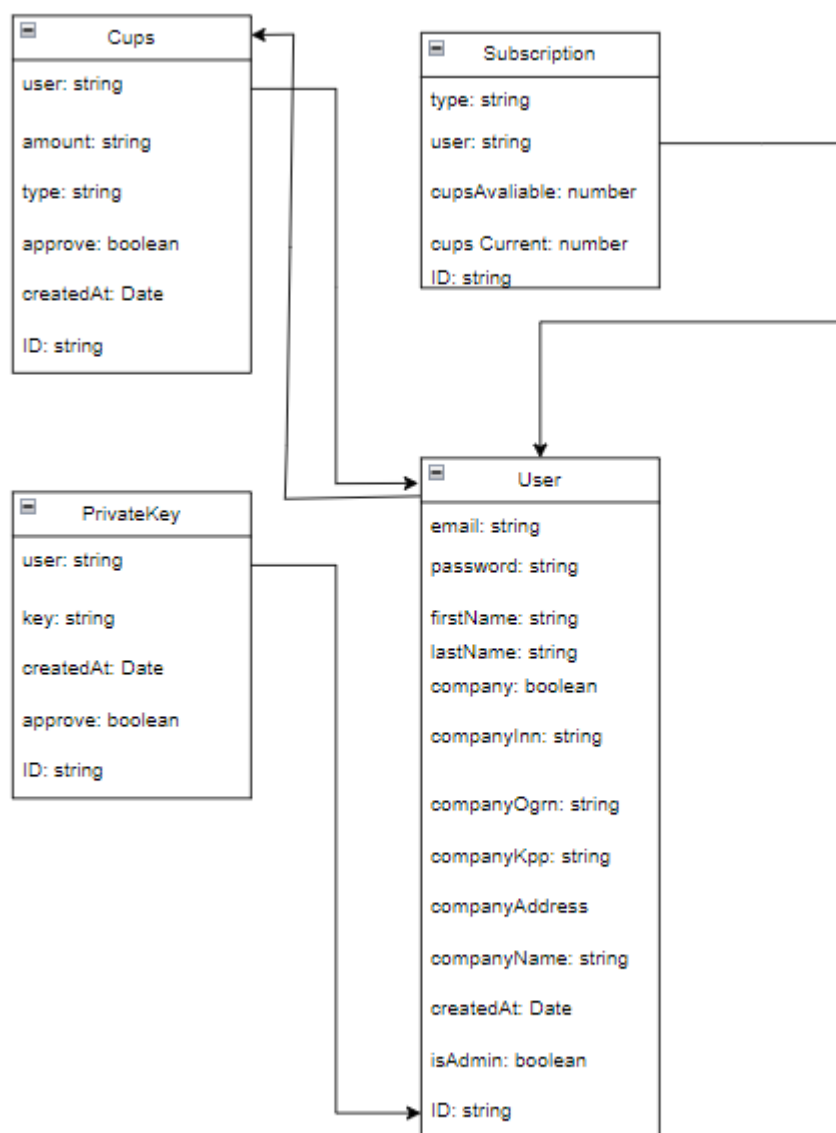


Рисунок 20 – Схема базы данных

На рисунке 21 представлен пример записи в базе данных о клиенте

```
_id: ObjectId('6656637ba73def8a5e0fa1a5')
email: "Kram123@work.com"
password: "$2a$10$3CsyY1wh6kE/wdvaNsYs10kQf/o4ep0R1CRiDndha3IpenKuOhmQm"
firstName: "Крамарь"
lastName: "Дмитрий"
company: true
companyInn: "0123456789"
companyOgrn: "1112223334445"
companyKpp: "123456789"
companyAddress: "Проспект Маршала Жукова 24"
companyName: "ООО \"Крам\""
isAdmin: false
createdAt: 2024-05-28T23:06:35.131+00:00
__v: 0
```

Рисунок 21 – Запись в базе данных

3 Разработка информационной системы

3.1 Разработка интерфейса

Проект планируется реализовать в три этапа:

1 Этап разработки и тестирования (7-10 месяцев).

Этот этап включает в себя разработку макета, программирование веб (frontend) и серверной (backend) частей, а также построение архитектуры и пробное тестирование.

2 Этап запуска и масштабирования (6-12 месяцев).

На этом этапе планируется развёртывание маркетинговой кампании, налаживание интеграций с кафе, налаживание партнёрских программ и последующий официальный запуск системы, и постепенное масштабирование.

3 Этап поддержки и оптимизации (после 12 месяцев).

Этот этап предполагает постоянное совершенствование функционала, осуществление поддержку пользователей, улучшение пользовательского опыта и расширение географии проекта.

Чтобы удовлетворить требования адаптивности и удобства главная страница сайта представляет собой лендинг, содержащий всю основную информацию о стартапе, также для удобства пользователя была разработана панель навигации, на которой лишь одна кнопка, представляющая собой логотип компании, она выполняет функцию «Вернуться на главную страницу». Пользовательский интерфейс лендинга представлен на рисунках 22-24.

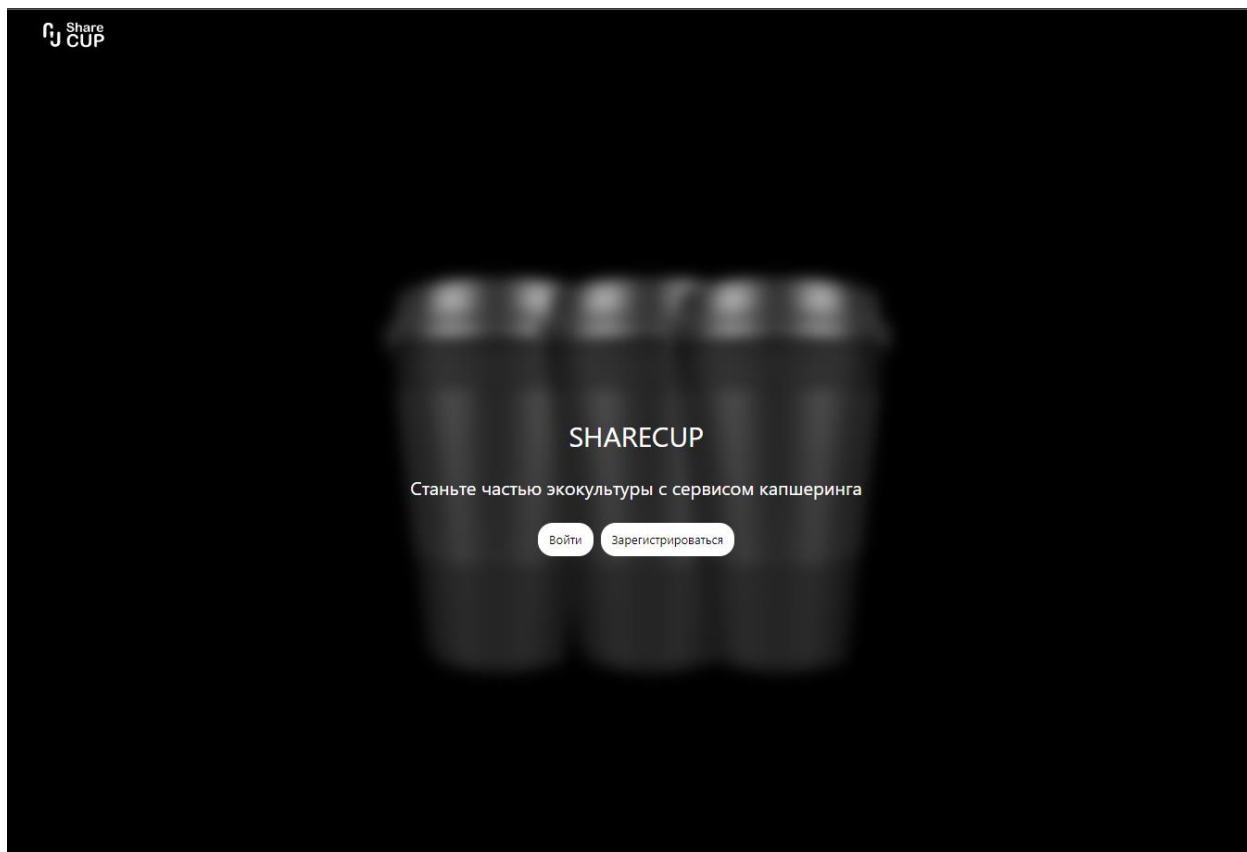


Рисунок 22 – Главная страница

Чуть ниже представлены доступные варианты подписок для физических и юридических лиц (пользователей).

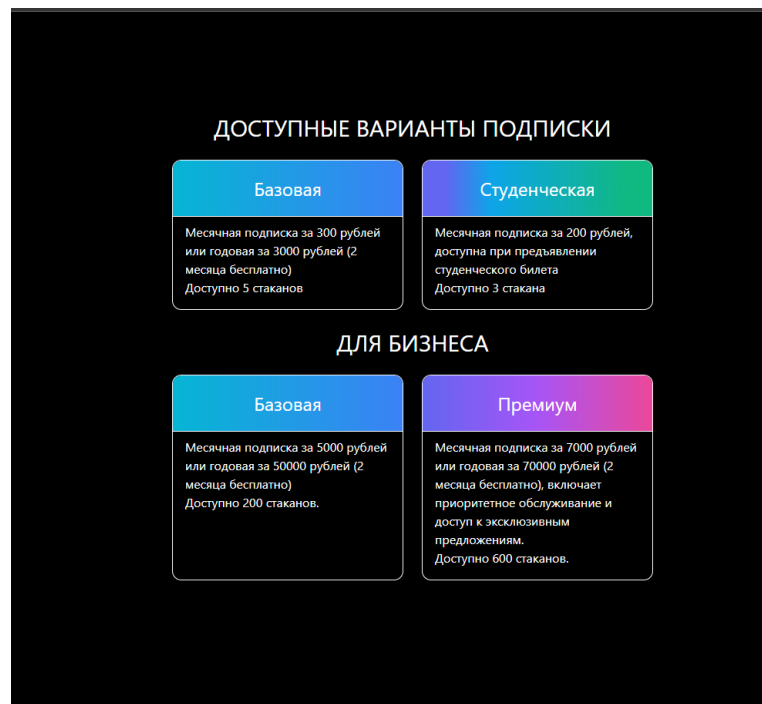


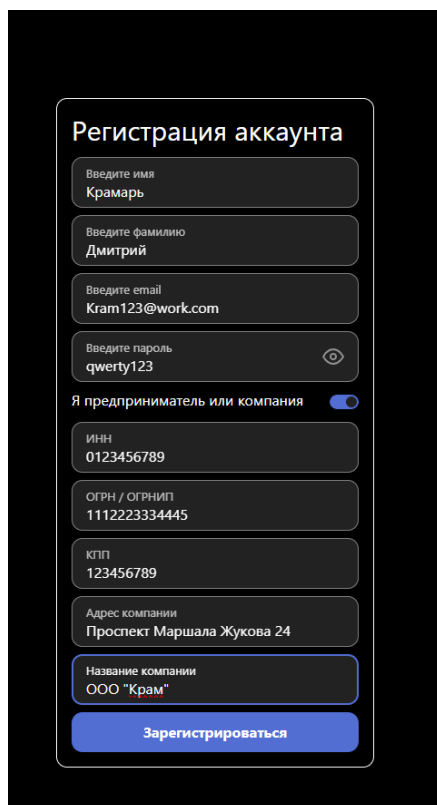
Рисунок 23 – Раздел подписок на главной странице

Еще ниже представлена основная информация о стартапе, отвечающая на вопросы: «Кто мы», «Что мы предлагаем», «Для чего мы это делаем».



Рисунок 24 – Раздел лендинга с основной информацией о стартапе

Рассмотрим пример регистрации аккаунта юридического лица, представленный на рисунках 25-29.



The image shows a registration form titled "Регистрация аккаунта" (Account Registration) on a dark background. The form contains several input fields with pre-filled data: "Введите имя" (Enter name) with "Крамарь", "Введите фамилию" (Enter surname) with "Дмитрий", "Введите email" (Enter email) with "Kram123@work.com", and "Введите пароль" (Enter password) with "qwerty123". There is a toggle switch labeled "Я предприниматель или компания" (I am an entrepreneur or company) which is turned on. Below this are fields for "ИНН" (0123456789), "ОГРН / ОГРНИП" (1112223334445), "КПП" (123456789), "Адрес компании" (Company address) with "Проспект Маршала Жукова 24", and "Название компании" (Company name) with "ООО 'Крам'". At the bottom is a blue button labeled "Зарегистрироваться" (Register).

Рисунок 25 – Регистрация аккаунта юридического лица

Сразу после этого действия в консоли программы была сформирована запись, которая переносится в базу данных.

```
"password": "qwerty123",
"firstName": "Крамарь",
"lastName": "Дмитрий",
"company": true,
"companyInn": "0123456789",
"companyOgrn": "1112223334445",
"companyKpp": "123456789",
"companyAddress": "Проспект Маршала Жукова 24",
"companyName": "ООО \"Крам\""
}
```

Рисунок 26 – Запись в БД

Войдя в аккаунт, предоставляется возможность изменения данных аккаунта, выход из сессии и кнопка установления подписки.




Рисунок 27 – Личный кабинет без подписки

Изменение аккаунта

Введите имя
Крамарь

Введите фамилию
Дмитрий

Введите email
Kram123@work.com

Введите пароль
qwerty2899 

ИНН
0123456789

ОГРН / ОГРНИП
1112223334445

КПП
123456789

Адрес компании
Проспект Маршала Жукова 24

Название компании
ООО "Крам"

Изменить

Рисунок 28 – Форма изменения данных

Установив подписку на месяц для юридических лиц веб-приложение переходит в состояние полноценной работы юридических лиц, при котором появляется возможность заказывать поставку или отгрузку стаканов, а также появляется возможность выдать или принять стакан у посетителя. В правом нижнем углу расположена история приемок и отгрузок.

The screenshot displays the 'ShareCUP' personal cabinet for a business user. At the top left is the 'ShareCUP' logo, and at the top right is a 'Выйти' (Logout) button. The main section shows the user's name 'Крамарь Дмитрий' and company 'ООО "Крам"', with an 'Изменить данные аккаунта' (Change account details) button. Below this, the subscription status is shown as 'Базовая для бизнеса (месячная)' (Basic for business (monthly)), with 'По подписке: 18600 стаканов' (By subscription: 18600 cups) and 'В использовании: 0 стаканов' (In use: 0 cups), accompanied by an 'Изменить подписку' (Change subscription) button. The interface is divided into several functional blocks: 'Передать стакан клиенту' (Transfer cup to client) and 'Забрать стакан у клиента' (Take cup from client), each with a text input for the client code and a corresponding action button; 'Заказать стаканы' (Order cups) and 'Отгрузить стаканы' (Load cups), each with a text input for the quantity and an action button. A large section on the right is titled 'История заказов / отгрузок' (Order / load history). The footer of the interface includes the text 'ShareCUP 2024'.

Рисунок 29 – Личный кабинет с подпиской

Рассмотрим примеры взаимодействия с информационной системой. Для начала поставим в кафе 5000 стаканов на ближайшую неделю, затем отгрузим 100 из них для прохождения обработки или переработки и последующего возвращения в цикл. Результат проделанных действий представлен на рисунках 30-31

ShareCUP

Выйти

Крамарь Дмитрий

ООО "Крам"

Изменить данные аккаунта

Базовая для бизнеса (месячная)

По подписке: 18600 стаканов

В использовании: 0 стаканов

Изменить подписку

Передать стакан клиенту

Введите код клиента

Передать

Забрать стакан у клиента

Введите код клиента

Забрать

Заказать стаканы

Введите кол-во стаканов

5000

Заказать

Стаканы успешно получены

Отгрузить стаканы

Введите кол-во стаканов

Отгрузить

История заказов / отгрузок

Заказано	5000 шт.
----------	----------

ShareCUP 2024

Рисунок 30 – Заказ 5000 стаканов

ShareCUP

Выйти

Крамарь Дмитрий

ООО "Крам"

Изменить данные аккаунта

Базовая для бизнеса (месячная)

По подписке: 18600 стаканов

В использовании: 0 стаканов

Изменить подписку

Передать стакан клиенту

Введите код клиента

Передать

Забрать стакан у клиента

Введите код клиента

Забрать

Заказать стаканы

Введите кол-во стаканов

5000

Заказать

Стаканы успешно получены

Отгрузить стаканы

Введите кол-во стаканов

100

Отгрузить

Стаканы успешно сданы

История заказов / отгрузок

Заказано	5000 шт.
Отгружено	100 шт.

ShareCUP 2024

Рисунок 31 – Отгрузка 100 стаканов

Далее рассмотрим интерфейс личного кабинета пользователя физического лица и передадим ему в пользование 3 стакана по коду. Результаты действий представлены на рисунках 32-33

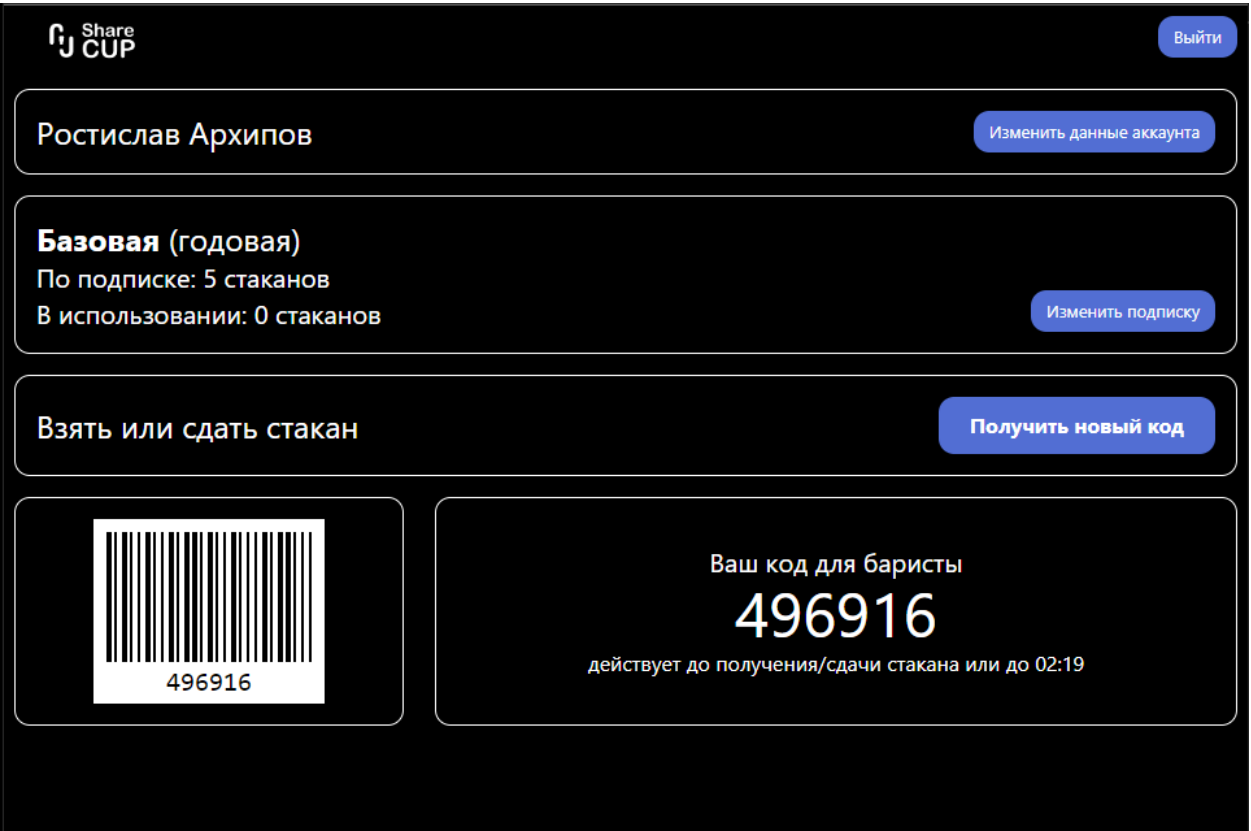


Рисунок 32 – Личный кабинет физического лица

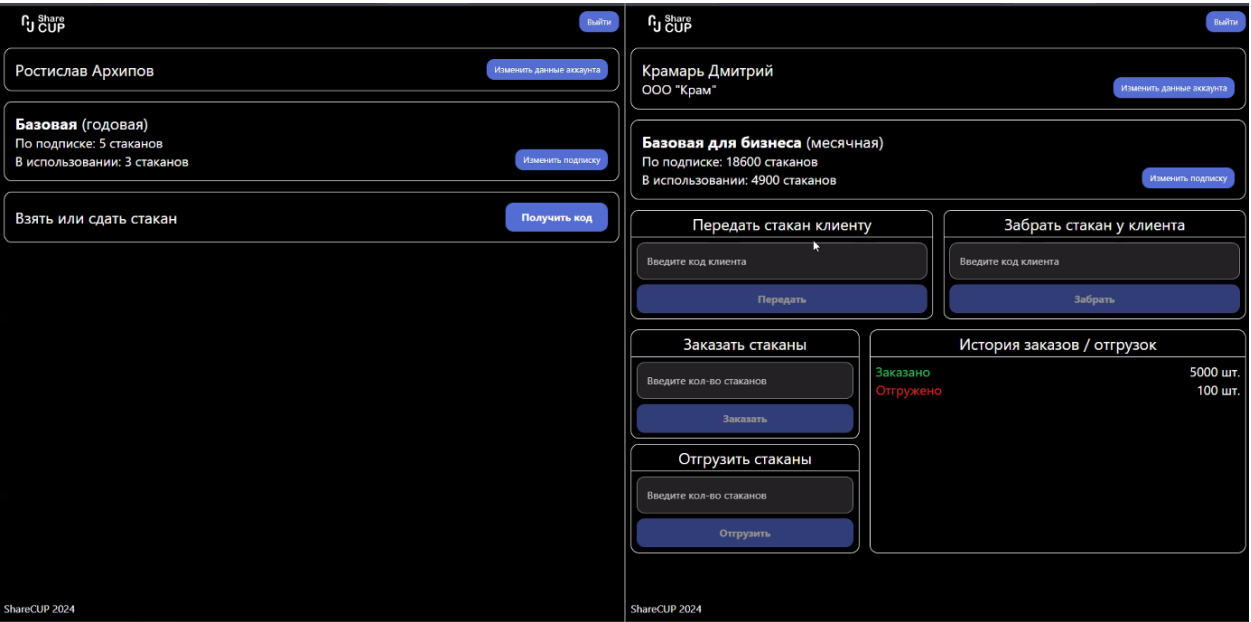
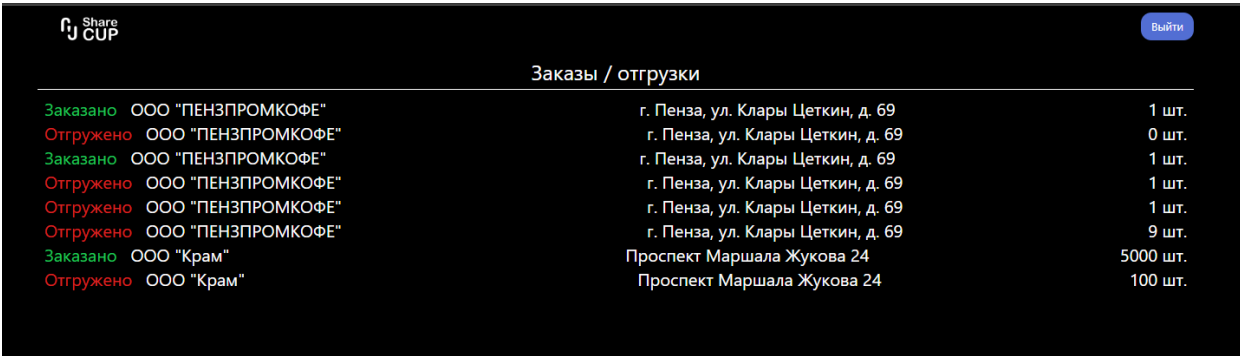


Рисунок 33 – Предоставление кафе в пользование физическому лицу
стаканов

На рисунке 34 представлен личный кабинет админ-пользователя, в котором доступна история всех заказов поставок и отгрузок для юридических лиц.



The screenshot shows a web interface for 'Share CUP' with a dark theme. At the top right is a 'Выйти' (Logout) button. The main section is titled 'Заказы / отгрузки' (Orders / Shipments). Below this is a table with three columns: status, company name, and address/quantity. The status column uses color coding: green for 'Заказано' (Ordered) and red for 'Отгружено' (Shipped). The table contains eight rows of data.

Заказы / отгрузки		
Заказано	ООО "ПЕНЗПРОМКОФЕ"	г. Пенза, ул. Клары Цеткин, д. 69 1 шт.
Отгружено	ООО "ПЕНЗПРОМКОФЕ"	г. Пенза, ул. Клары Цеткин, д. 69 0 шт.
Заказано	ООО "ПЕНЗПРОМКОФЕ"	г. Пенза, ул. Клары Цеткин, д. 69 1 шт.
Отгружено	ООО "ПЕНЗПРОМКОФЕ"	г. Пенза, ул. Клары Цеткин, д. 69 1 шт.
Отгружено	ООО "ПЕНЗПРОМКОФЕ"	г. Пенза, ул. Клары Цеткин, д. 69 1 шт.
Отгружено	ООО "ПЕНЗПРОМКОФЕ"	г. Пенза, ул. Клары Цеткин, д. 69 9 шт.
Заказано	ООО "Крам"	Проспект Маршала Жукова 24 5000 шт.
Отгружено	ООО "Крам"	Проспект Маршала Жукова 24 100 шт.

Рисунок 34 – Админ – панель

3.2 Разработка API

Для разработки backend сервиса были использованы следующие технологии:

JavaScript является языком программирования с низким порогом вхождения. Низкоуровневый доступ к памяти или процессору в нём не предоставляется, так как изначально был разработан для работы в браузерах, где такие возможности не требуются. JavaScript довольно прост для изучения, что способствовало его быстрой популяризации. Возможности этого языка значительно зависят от среды выполнения. [9] Например, в Node.js JavaScript позволяет читать и записывать файлы, выполнять сетевые запросы и другие операции [11]. В браузере JavaScript обеспечивает доступ ко всем функциям, связанным с манипуляцией веб-страницами, взаимодействием с пользователями и веб-серверами [12].

TypeScript представляет собой расширение для языка JavaScript, устраняющее множество его недостатков и внедряющее строгую типизацию кода. Код на TypeScript выглядит схоже с JavaScript, но включает некоторые дополнения и изменения. Освоить TypeScript несложно, если пользователь уже знаком с классическим JavaScript. Код на TypeScript компилируется в JavaScript и подходит для разработки любых проектов в любых браузерах,

причём можно выбрать версию JavaScript, в которую будет компилироваться код.

«TypeScript - проект с открытым исходным кодом, поэтому он очень быстро развивается. Многое, что появляется в TS, позже переходит и в JavaScript: например, let и const, стрелочные функции и так далее» [13].

Клиент-серверное взаимодействие было осуществлено в стиле REST (Representational State Transfer) — это способ создания API с помощью протокола HTTP [14]. Каждый объект на сервере в этом протоколе имеет свой уникальный URL-адрес в строгом последовательном формате [15].

Полноценный API-сервис разрабатывается при помощи Node.js и основанного на нем фреймворка Nest. Node позволяет разработчику использовать JavaScript для написания серверного ПО и дает преимущества благодаря своему асинхронному подходу к обработке операций [16][17].

REST API предоставляет 4 метода взаимодействия по протоколу HTTP с объектами серверного приложения:

- GET метод – предназначен для чтения информации. GET-запросы возвращают данные с сервера;
- POST – создание новых записей;
- PUT – редактирование записей;
- DELETE - удаление записей [18][19][20].

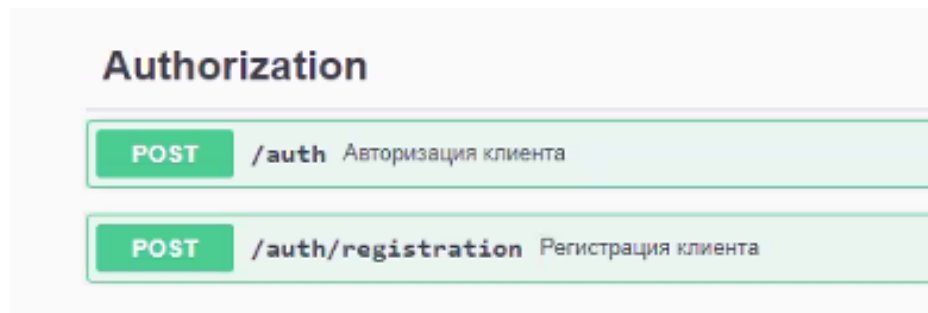
Данный набор запросов имеет название CRUD, что является аббревиатурой: create, read, update, delete. Этот набор предусматривает стандартные функции по манипуляции над данными [21].

Этими запросами были разработаны входные точки для манипуляции с данными при помощи Swagger.

Swagger, сейчас известный как OpenAPI Specification, это инструмент, предназначенный для проектирования, создания, документирования и использования RESTful веб-сервисов. Главная цель swagger – обеспечить стандартизацию способов описания API, что позволит

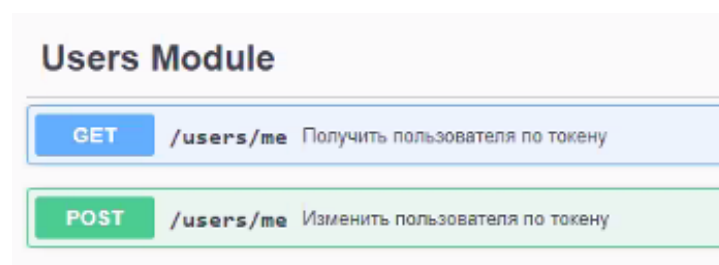
разработчикам и компьютерным системам легко понимать возможности сервиса без доступа к коду [22][23].

Наборы входных точек представлены на рисунках 35-38



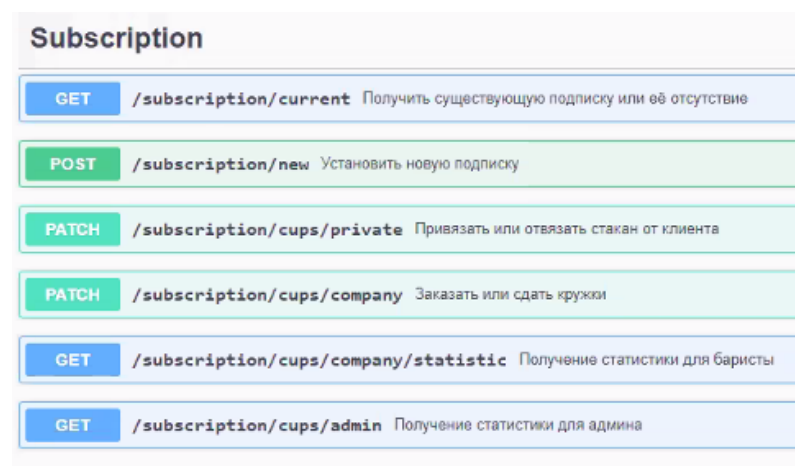
Authorization	
POST	/auth Авторизация клиента
POST	/auth/registration Регистрация клиента

Рисунок 35 – Запросы при авторизации/регистрации



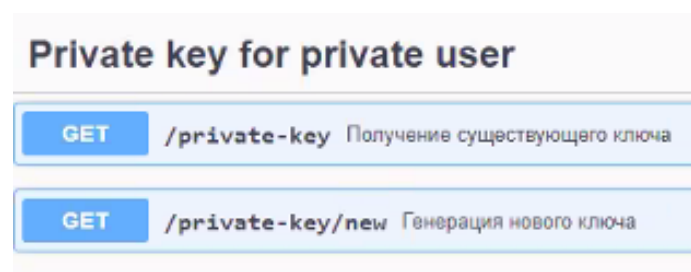
Users Module	
GET	/users/me Получить пользователя по токену
POST	/users/me Изменить пользователя по токену

Рисунок 36 – Запросы по JWT-токену



Subscription	
GET	/subscription/current Получить существующую подписку или её отсутствие
POST	/subscription/new Установить новую подписку
PATCH	/subscription/cups/private Привязать или отвязать стакан от клиента
PATCH	/subscription/cups/company Заказать или сдать кружки
GET	/subscription/cups/company/statistic Получение статистики для баристы
GET	/subscription/cups/admin Получение статистики для админа

Рисунок 37 – Запросы для личного кабинета



Private key for private user	
GET	/private-key Получение существующего ключа
GET	/private-key/new Генерация нового ключа

Рисунок 38 – Запросы для получения ключа получения стакана

На рисунке 39 представлены модели данных DTO (Data Transfer Object). Это паттерны проектирования, используемые для передачи данных между

слоями веб-приложения через сетевые соединения. Основа DTO – это упаковка данных в объекты, их инкапсуляция и абстракция, всё это необходимо для того, чтобы оптимизировать структуру передачи данных, ускорить процесс обмена данными путем сокращения количества вызовов методов, запросов и ответов между клиентом и сервером. В контексте веб-разработки сервер может использовать DTO для отправки всех данных, необходимых для отображения веб-страницы, в одном ответе [24].



Рисунок 39 – Модели передаваемых между слоями данных

4 Финансовый план и оценка проекта

4.1 Финансовый план разработки ИС

Для того, чтобы оценить финансовый план разработки информационной системы в среде MS Project была создана диаграмма Ганта, в которой расписан весь спектр работ. Всего их 6:

1 Составление ТЗ и начало работ

Этап состоит из сбора информации о качествах ИС, разработке функциональных и нефункциональных требований, а также утверждении технических требований к проекту

Ответственные: Системный/Бизнес аналитик, SEO

2 Техническое проектирование

В этом этапе производится разработка макета веб-ориентированной системы и его архитектуры.

Ответственные: Дизайнер, Backend и Frontend разработчики

3 Разработка программной части

Данный этап представляет собой программирование frontend и backend частей, что подразумевает разработку интерфейса и серверной логики, создание функционала ПО и разработку базы данных, а также последующее объединение всех частей.

Ответственные: Backend разработчики, Frontend разработчики, Дизайнер

4 Предварительное тестирование

На данном этапе производится автоматическое и комплексное тестирование, проверяется стабильность и производительность системы

Ответственные: Тестировщики

5 Опытная эксплуатация и оптимизация

Подготовка к опытной эксплуатации и проведение испытаний – это завершающий шаг перед запуском системы, включающий финальные корректировки.

Ответственные: Тестировщики, Backend и Frontend разработчики

6 Поддержка и документация

Завершающий этап – создание обучающей документации и подготовка к поддержке пользователей.

Ответственные: Системный/Бизнес аналитик, SEO

На рисунках 40-41 представлена полученная диаграмма Ганта. На рисунке 42 представлен получившийся лист ресурсов

название	длительность	начало	преддеств	название ресурса
Система скоринга и выдачи кредита	213,26 дней	Чт 01.02.24		
Начало работ	0 дней	Чт 01.02.24		
Составление ТЗ	17 дней	Чт 01.02.24		
Сбор информации о качествах ИС, нужных для стартапа	10 дней	Чт 01.02.24	2	SEO;Системный/Бизнес аналитик[200%]
Разработка функциональных и нефункциональных требований	6 дней	Чт 15.02.24	4	SEO[150%];Системный/Бизнес аналитик
Утверждение ТЗ	1 день	Пт 23.02.24	5	SEO[150%]
Техническое проектирование	34,5 дней	Пн 26.02.24	6	
Разработка макета веб-ориентированной системы	11,83 дней	Пн 26.02.24		Дизайнер[150%];Frontend dev
Разработка архитектуры веб-ориентированной ИС	13 дней	Вт 12.03.24	8	Backend dev;Frontend dev
Разработка дизайна веб-ориентированной ИС	8,67 дней	Пт 29.03.24	9	Дизайнер[150%]
Утверждение технических требований к проекту	1 день	Чт 11.04.24	10	SEO
Разработка программной части	95,71 дней	Пт 12.04.24		
Frontend разработка	20 дней	Пт 12.04.24	11	Frontend dev
Backend разработка	17 дней	Пт 12.04.24	11	Backend dev
Соединение функций ПО и разработанного графического интерфейса	25,71 дней	Пт 10.05.24	14;13	Backend dev;Frontend dev;Дизайнер
Разработка базы данных	20 дней	Пн 17.06.24	15	Backend dev;Системный/Бизнес аналитик
Написание xml/json для вывода из веб в БД	30 дней	Пн 15.07.24	16	Backend dev;Frontend dev
Предварительные испытания	18,67 дней	Пн 26.08.24	17	
Автономные испытания	8 дней	Пн 26.08.24		Тестировщик[250%]
Комплексные испытания	8 дней	Чт 05.09.24	19	Тестировщик[250%]
Доработка и фикс багов	2,67 дней	Вт 17.09.24	20	Backend dev;Frontend dev
Опытная эксплуатация	39 дней	Чт 19.09.24		
Подготовка к опытной эксплуатации	5 дней	Чт 19.09.24	21	Тестировщик
Проведение опытной эксплуатации	24 дней	Чт 26.09.24	23	Тестировщик[250%]
Доработка и устранение недостатков	10 дней	Ср 30.10.24	24	Backend dev;Frontend dev
Промышленная эксплуатация	8 дней	Ср 13.11.24		
Создание обучающей документации	7 дней	Ср 13.11.24	25	Системный/Бизнес аналитик
Оформление гарантийной поддержки	1 день	Пт 22.11.24	27	SEO;Системный/Бизнес аналитик
Завершение работ	0 дней	Пн 25.11.24	28	

Рисунок 40 – План работ

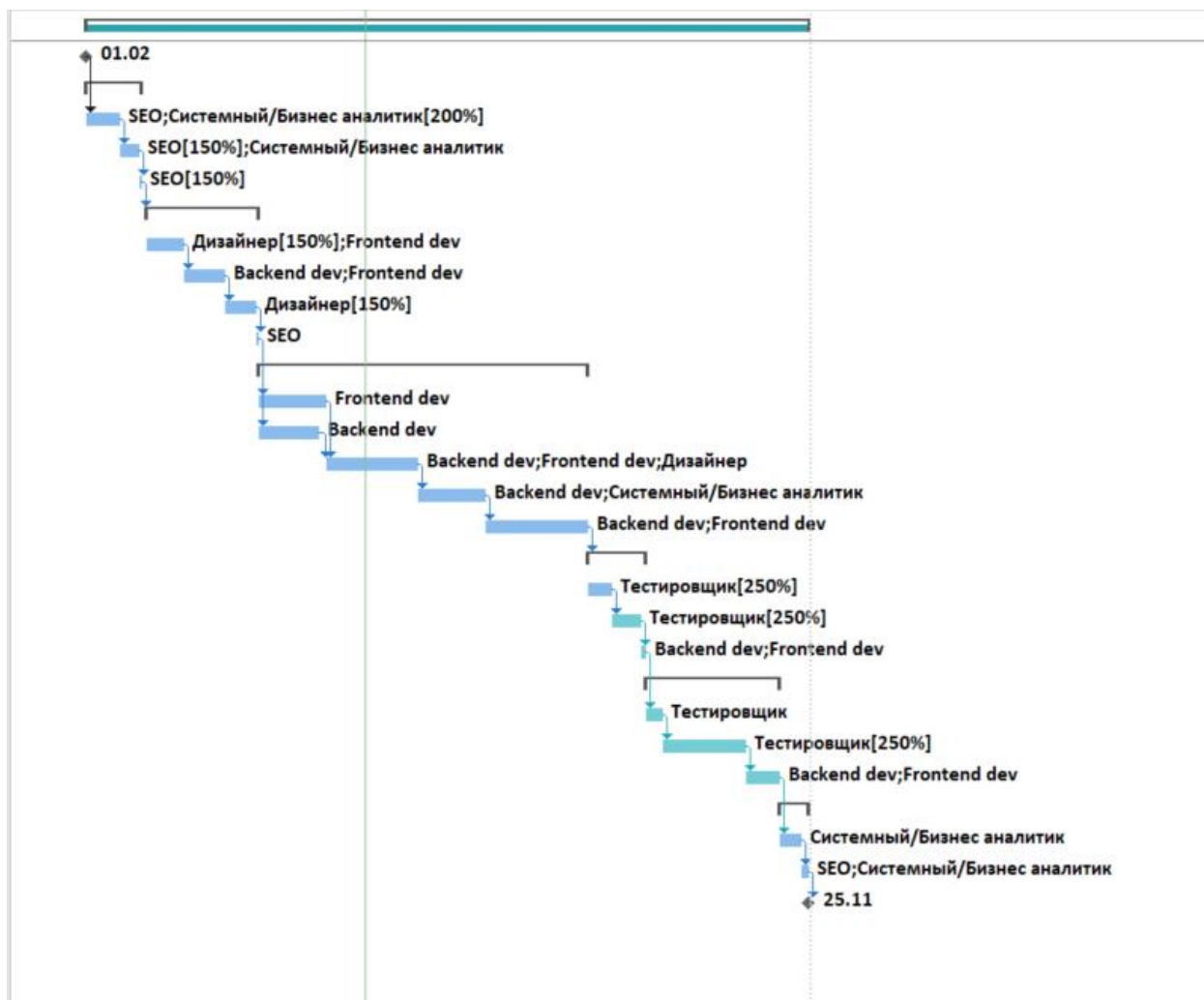


Рисунок 41 – Диаграмма Ганта

1	Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на исполн.	Начисление	Базовый календарь
	SEO	Трудовой		S	Менеджмент	100%	200 000,00Р/месяц	15 000,00Р/месяц	0,00Р	Пропорциональное	Standard
	Frontend dev	Трудовой		F	Проектная	350%	80 000,00Р/месяц	10 000,00Р/месяц	0,00Р	Пропорциональное	Standard
	Backend dev	Трудовой		B	Проектная	200%	100 000,00Р/месяц	10 000,00Р/месяц	0,00Р	Пропорциональное	Standard
	Системный/Бизнес аналитик	Трудовой		C	Проектная	200%	80 000,00Р/месяц	3 000,00Р/месяц	0,00Р	Пропорциональное	Standard
	Дизайнер	Трудовой		D	Проектная	150%	100 000,00Р/месяц	0,00Р/месяц	0,00Р	Пропорциональное	Standard
	Тестировщик	Трудовой		T	Проектная	400%	50 000,00Р/месяц	5 000,00Р/месяц	0,00Р	Пропорциональное	Standard

Рисунок 42 – Лист ресурсов

На рисунке 43 представлено графическое отображение движения денежных средств, составленное автоматически средой MS Project по разработанной диаграмме Ганта. Исходя из неё можно утверждать о цене проектирования и разработки полноценной информационной системы, её тестирования и поддержки.

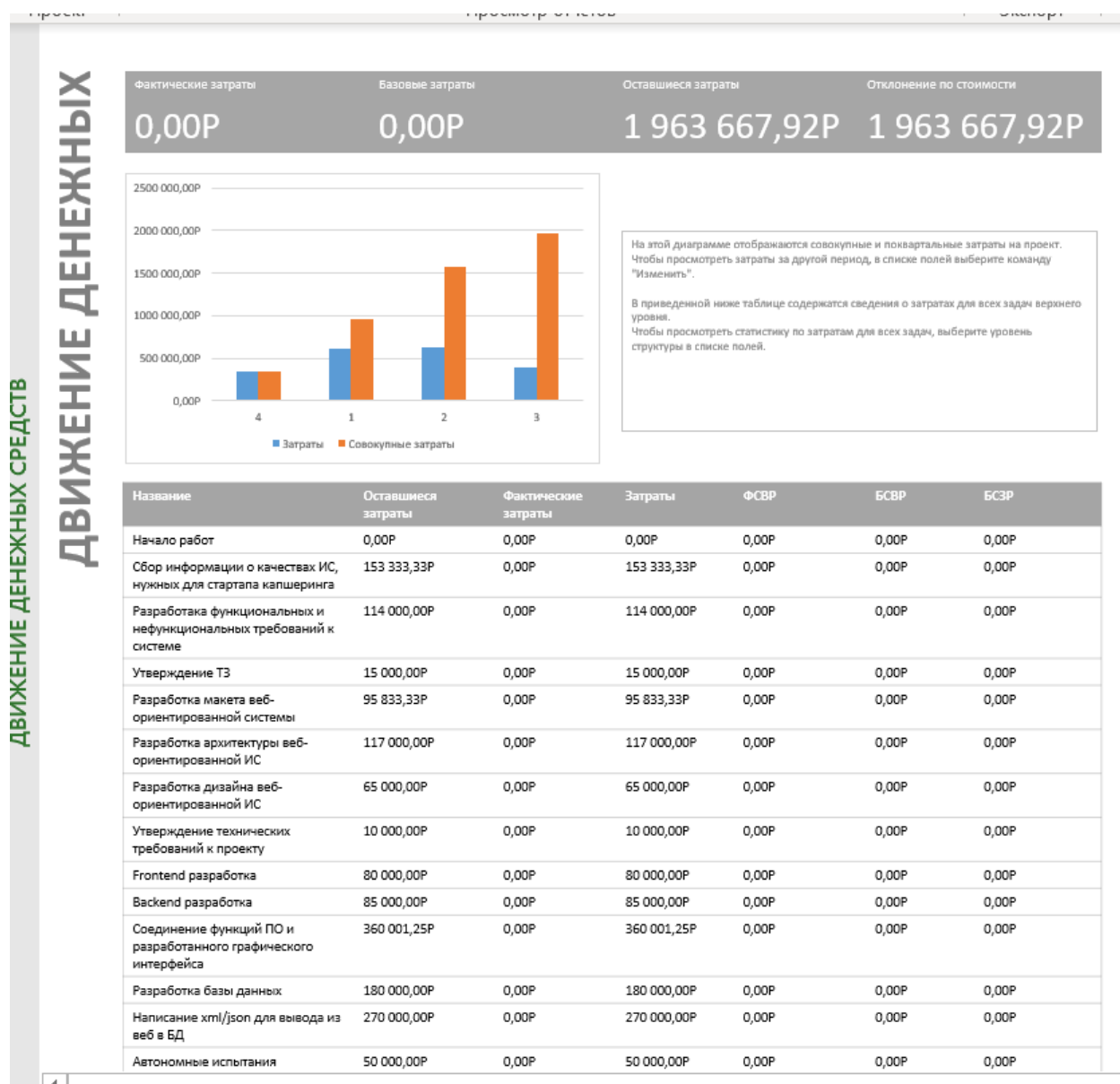


Рисунок 43 – Графическое отображение движения денежных средств

4.2 Финансовый план проекта

Целью любого предприятия в рамках рыночной экономики является получение прибыли, способной обеспечить ее дальнейшее развитие.

Для того, чтобы посчитать рентабельность проекта, следует сначала определить стартовые затраты. В случае со стартапом капшеринга стартовыми затратами будут являться регистрация бизнеса – 10000 рублей и лицензия на торговую деятельность 7500 рублей.

Далее нужно определить фонд оплаты труда наемных сотрудников (ФОТ), представленный в таблице 4.

Таблица 4 – Фонд оплаты труда

Наименование позиции	Затраты руб.
Маркетолог	60000
Водитель	30000
Грузчик	35000
Бухгалтер	70000
Администратор ИС	50000
Юрист	35000

Далее нужно определить основные потоки прибыли и переменные издержки, представлены в таблице 5. В этой модели будем исходить из предположения о том, что одно подписанное кафе приносит 70 клиентов физических лиц, из которых 80% пользуются студенческой скидкой. Что касается подписок юридических лиц, будем считать, что из них 20% используют премиум подписку. На одно кафе в среднем

Таблица 5 – Переменные издержки и основные потоки прибыли

Подписка	Итого руб./мес.		Затраты/1 цикл
Физическое лицо	300	21000	4000
Юридическое лицо	5000	5000	2000 (обработка)
		26000	70000 (ФОТ)

Рассмотрим таблицу 6, в которой представлены затраты при 20 циклах заказов от юридических лиц. С помощью изменений количества юридических и физических лиц можно найти точку безубыточности предприятия.

Таблица 6 – Затраты и прибыль при 20 заказах в месяц

Доставка	3 поездки на партию (забрать со склада, отвезти в кафе, забрать для обработки)	50000 руб.	5 кафе	27000 (прибыль с 5 кафе)
Обработка	20 заявок	10000 руб.	350 физ. лиц	77000 (прибыль от подписок физ. лиц)
Утилизация	Эквивалент 1% заказов	500 руб.	355500 руб. (все затраты)	104000 руб. (прибыль за месяц)
ФОТ	Общий ФОТ	280000 руб.		
Аренда склада	5000 руб./мес.	5000 руб.		
Прочие расходы	10000 руб./мес	10000 руб.		

Исходя из данных таблицы, если разделить всю прибыль на все затраты получим сколько мы зарабатываем с одного рубля вложений и эта сумма будет равной 0,2925 рубля.

Рассмотрим таблицу 7, в которой произведены все те же расчёты, однако в большем масштабе, предположим, что система ShareCUP привлекла внимание 34 кафе, которые привели с собой 2380 физических пользователей.

Таблица 7 – Затраты и прибыль при 136 заказах в месяц

Доставка	3 поездки на партию	340000 руб.	34 кафе	183600 (прибыль с 34 кафе)
Обработка	136 заявок	68000 руб.	2380 физ. лиц	523600 (прибыль от подписок физ. лиц)
Утилизация	Эквивалент 1% заказов	500 руб.	703500 руб. (все затраты)	707200 руб. (прибыль за месяц)
ФОТ	Общий ФОТ	280000 руб.		
Аренда склада	5000 руб./мес.	5000 руб.		
Прочие расходы	10000 руб./мес	10000 руб.		

В данном случае, разделив все затраты на всю прибыль получим показатель 1,00525 рубля на рубль вложений. Показатели, при которых достигается такой результат, а именно 136 циклов на 34 кафе и 2380

физических лиц – есть показатели, при которых предприятие выходит на точку безубыточности.

Для оценки величины денежных средств, которые ожидается получить от проекта и срока его окупаемости используем подсчёт NPV.

NPV – это суммарная стоимость денежных потоков на определенный момент времени жизненного цикла проекта. Иными словами, это будущий финансовый результат в эквиваленте суммы на текущий момент. Аббревиатура расшифровывается как Net Present Value (чистая приведенная стоимость) [25].

Классическая формула расчета чистой приведенной стоимости выглядит так – формула (2):

$$NPV = \sum_{k=1}^n \frac{P_k}{(1+i)^n} - \sum_{k=1}^m \frac{IC_j}{(1+i)^j} \quad (2)$$

Где:

n – количество лет до инвестиций;

m – количество лет после инвестиций;

j – разница между m и n .

Для его подсчёта используем онлайн калькулятор и данные из расчётов точки безубыточности. Предположим, что сумма инвестиций в проект на стадии его запуска составит 1500000 рублей, а ставка дисконтирования будет равной ключевой ставке центрального банка, то есть 16%. Результаты расчёта представлены на рисунке 44.

Год	Поток денежной наличности, тыс.руб.		Ставка дисконта $r = 16.0$		
	Приток	Отток	Чистый поток	Дисконт	NPV, тыс.руб.
0 (нач.года 1)	1500000		1500000	1	1500000
1 (кон.года 1)	104000	355500	-251500	0.862	-216793
2 (кон.года 2)	728000	715500	12500	0.743	9287.5
3 (кон.года 3)	1248000	1016000	232000	0.641	148712
4 (кон.года 4)	2080000	1497000	583000	0.552	321816
5 (кон.года 5)	0	0	0	0	0
6 (кон.года 6)	0	0	0	0	0
7 (кон.года 7)	0	0	0	0	0
8 (кон.года 8)	0	0	0	0	0
9 (кон.года 9)	0	0	0	0	0
10(кон.года 10)	0	0	0	0	0
Итого	5660000	3584000	2076000	NPV 1763022.5	тыс.руб.

Рисунок 44 – Расчёт NPV по потокам денежных средств

Для оценки были предположены темпы роста: в первый расчётный период количество подписанных кафе составит 5, во второй – 35, в третий – 60 и в четвертый – 100 кафе.

При таких данных роста числа подписанных клиентов срок окупаемости проекта по модели с высокой степенью абстракции без учёта разработки информационной системы составит 4 расчётных периода, с учётом разработки, учитывая потенциал масштабируемости около 7 расчётных периодов.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы была разработана информационная система стартапа капшеринга многоразовых стаканов. Подобного рода стартапы способствуют формированию экологической культуры и практики повторного использования ресурсов, что является важным шагом к экологически устойчивому потреблению.

Разработанная в работе маркетинговая стратегия, направленная на привлечение первых пользователей и партнёров из среды малых бизнесов сферы общественного питания, позволит продемонстрировать востребованность предложения и его конкурентоспособность.

Проект выпускной квалификационной работы демонстрирует значительный потенциал для масштабирования, особенно с учётом текущих трендов экологических инноваций. Дальнейшие исследования и разработка предполагают расширение географии сервиса и внедрение дополнительных функций для повышения удобства пользователей. Проведенные в 4 главе расчеты позволяют сделать выводы о том, что проект окупится в течение 7 месяцев.

Анализ теоретических основ и практическое применение современных технологий разработки позволило создать систему, которая удовлетворяет текущим требованиям к подобного рода проектам. Разработанная система включает все необходимые компоненты и функции для эффективного взаимодействия с пользователями и управления процессами аренды и возврата многоразовых стаканов, что создает стартапу ShareCUP конкурентное преимущество перед всеми подобными решениями на отечественном рынке.

Таким образом, можно утверждать об успешной реализации стартапа и выполнении всех поставленных целей и задач. Опыт, полученный в ходе работы по проектированию и разработке информационной системы, станет основой для будущих исследований в данной области.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лебедева Д.Р. Человек экологический: повседневные практики заботы об окружающей среде как атрибут современного субъекта в представлениях молодых москвичей // Журнал социологии и социальной антропологии. 2021 24(2): 110–143. URL: <https://doi.org/10.31119/jssa.2021.24.2.5> (дата обращения 01.06.2024).
2. Геологические аспекты глобализации // Cyberleninka [Электронный ресурс]. Режим доступа: <https://cyberleninka.ru/article/n/geoekologicheskie-aspekty-globalizatsii/viewer> (дата обращения 29.05.2024).
3. Основные принципы и приоритетные направления государственной политики в области обращения с отходами // КонсультантПлюс [Электронный ресурс]. URL: https://www.consultant.ru/document/cons_doc_LAW_19109/f380561eb65d28708f522e4230771b49d1d5eb4e/ (дата обращения 29.05.2024).
4. Федеральный закон "Об отходах производства и потребления" от 24.06.1998 N 89-ФЗ (последняя редакция) // КонсультантПлюс [Электронный ресурс]. URL: https://www.consultant.ru/document/cons_doc_LAW_19109/f380561eb65d28708f522e4230771b49d1d5eb4e/ (дата обращения 03.06.2024).
5. Целевая аудитория мероприятия: как определить и привлечь идеальную целевую аудиторию мероприятия для роста бизнеса // FasterCapital [Электронный ресурс]. URL: <https://fastercapital.com/ru/content/%D0%A6%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F-%D0%B0%D1%83%D0%B4%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F-%D0%BC%D0%B5%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D1%8F%D1%82%D0%B8%D1%8F--%D0%BA%D0%B0%D0%BA-%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B8>

[%D1%82%D1%8C-%D0%B8-%D0%BF%D1%80%D0%B8%D0%B2%D0%BB%D0%B5%D1%87%D1%8C-%D0%B8%D0%B4%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D1%83%D1%8E-%D1%86%D0%B5%D0%BB%D0%B5%D0%B2%D1%83%D1%8E-%D0%B0%D1%83%D0%B4%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8E-%D0%BC%D0%B5%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D1%8F%D1%82%D0%B8%D1%8F-%D0%B4%D0%BB%D1%8F-%D1%80%D0%BE%D1%81%D1%82%D0%B0-%D0%B1%D0%B8%D0%B7%D0%BD%D0%B5%D1%81%D0%B0.html](#) (дата обращения 28.05.2024).

6. Наумов И. А. Модель бизнеса по подписке в современных реалиях // Бизнес-образование в экономике знаний [Электронный ресурс]. URL: <https://cyberleninka.ru/article/n/model-biznesa-po-podpiske-v-sovremennyh-realiyah/viewer> (дата обращения 29.05.2024).

7. Сколько кофе пьют россияне // TinkoffJournal [Электронный ресурс]. URL: <https://journal.tinkoff.ru/coffee-stat/> (29.05.2024).

8. MongoDB. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94%D0%B8%D0%B7%D0%BD%D0%B5%D1%81%D0%B0%D0%B8%D0%B7%D0%BD%D0%B5%D1%81%D0%B0.html (дата обращения 31.05.2024).

9. Документно-ориентированная база данных [Электронный ресурс]. URL: <https://tproger.ru/translations/types-of-nosql-db/> (дата обращения 30.05.2024).

10. Марейн Хавербеке. Выразительный JavaScript. Современное веб-программирование.: Питер, 2022.

11. Дэвид Флэнаган. JavaScript. Подробное руководство.: Диалектика-Вильямс, 2021.

12. JWT [Электронный ресурс]. URL: <https://jwt.io/introduction> (дата обращения 31.05.2024).
13. TypeScript. [Электронный ресурс]. URL: https://skillbox.ru/media/code/typescript_kak_s_nim_rabotat_i_chem_on_otlichaetsya_ot_javascript (дата обращения 01.06.2024).
14. NodeJs // Skillfactory [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/node-js/> (дата обращения 29.05.2024).
15. Алекс Янг, Майк Кантелон. Node.js в действии.: Питер, 2018.
16. Шелли Пауэрс. Изучаем Node. Переходим на сторону сервера.: Питер, 2017.
17. Хэррон Д. NODE.JS РАЗРАБОТКА СЕРВЕРНЫХ ВЕБ-ПРИЛОЖЕНИЙ НА JAVASCRIPT.: ДМК Пресс, 2022.
18. Роберт Мартин. Чистый код. Создание, анализ и рефакторинг.: Питер, 2010.
19. Nest Фреймворк [Электронный ресурс]. URL: <https://docs.nestjs.com/> (дата обращения: 01.06.2024).
20. Лучшие Node фреймворки [Электронный ресурс]. URL: <https://techrocks.ru/2021/04/15/node-js-frameworks-2021/> (дата обращения 31.05.2024).
21. Что такое CRUD-приложение и для чего он нужен // Habr Q&A [Электронный ресурс]. URL: <https://qna.habr.com/q/538933> (дата обращения 30.05.2024).
22. OpenAPI/Swagger для начинающих // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/776538/> (дата обращения 30.05.2024)
23. Swagger [Электронный ресурс]. URL: <https://evilinside.ru/chto-takoe-swagger/> (дата обращения 01.06.2024).
24. DTO в JS // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/567040/> (дата обращения 31.05.2024).

25. NPV - что это, как рассчитать по формуле [Электронный ресурс].

URL: <https://retireearly.ru/financial-literacy/npv-chistaja-privedennaja-stoimost>


(дата обращения 01.06.2024).

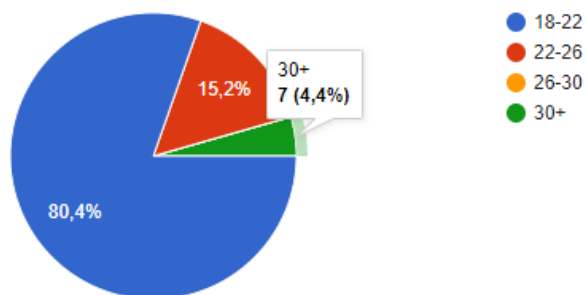
Приложения

Приложение А Результаты анкетирования (справочное)

Сколько Вам лет?


158 ответов

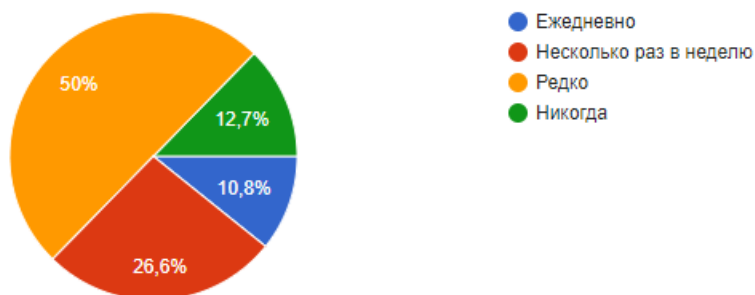
 Копировать



Как часто вы покупаете напитки в одноразовых стаканах?


158 ответов

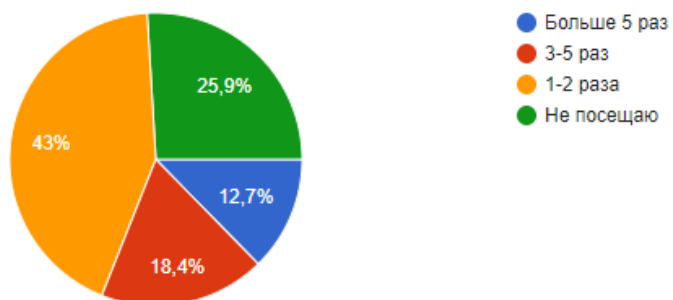
 Копировать




Сколько раз в неделю вы посещаете кафе или кофейни?

158 ответов

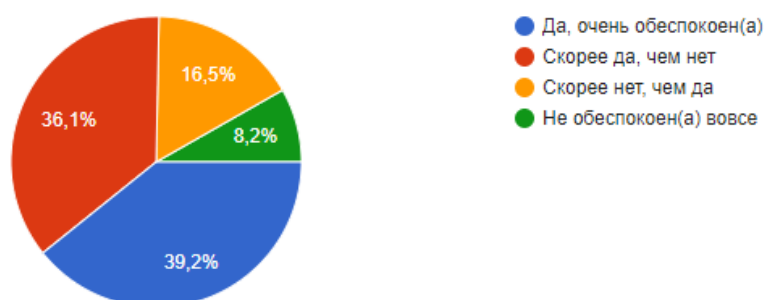
 Копировать




Были ли вы когда-нибудь обеспокоены количеством отходов от одноразовой посуды?

 Копировать

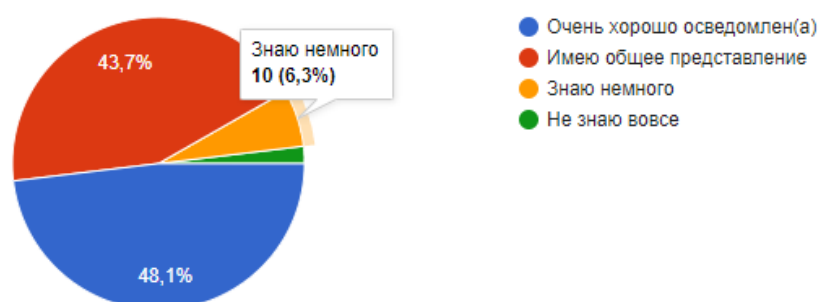
158 ответов



Знаете ли вы о вреде пластиковых отходов для окружающей среды?

 Копировать

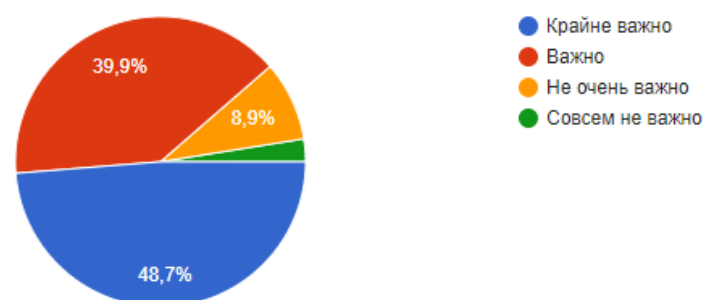
158 ответов



Считаете ли вы важным использование экологически чистых продуктов и услуг?

 Копировать

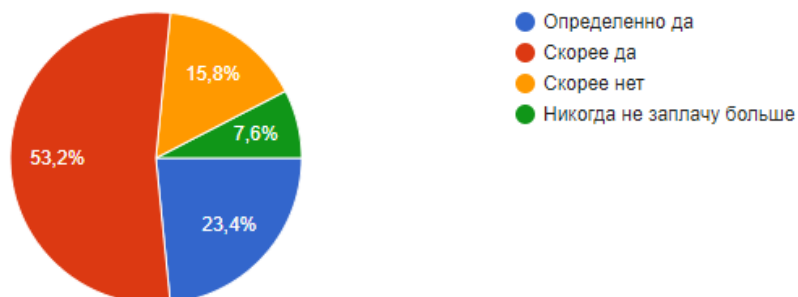
158 ответов



Готовы ли вы платить немного больше за экологически чистые решения?

 Копировать

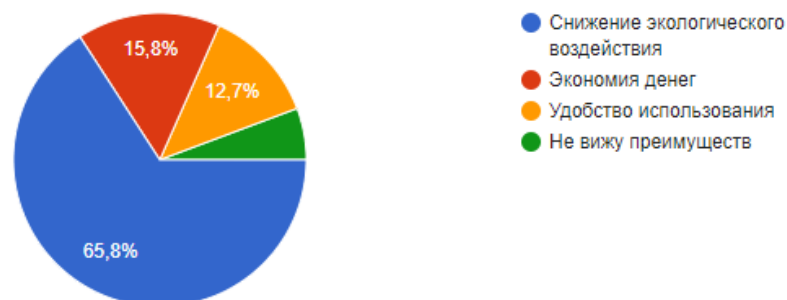
158 ответов




Какие преимущества вы видите в использовании многоразовых стаканов?

 Копировать

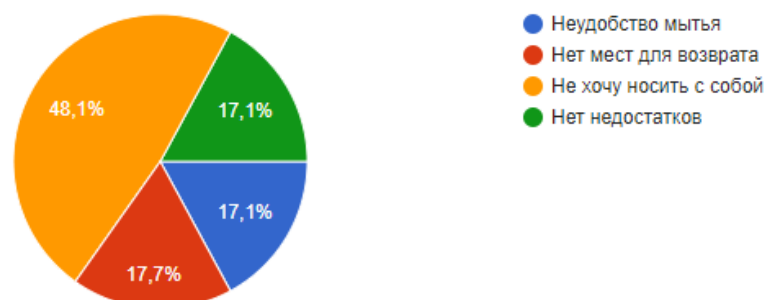
158 ответов




Какие недостатки могут удержать вас от использования многоразовых стаканов?

 Копировать

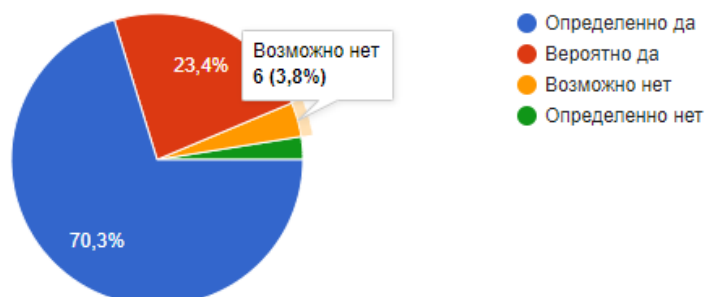
158 ответов



Предпочли бы вы использовать многоразовый стакан, если бы это было удобно и доступно?

 Копировать

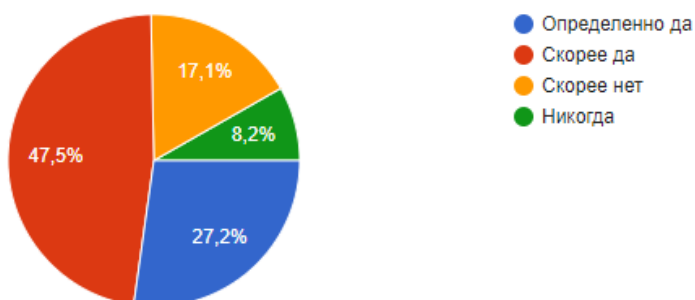
158 ответов



Были бы вы готовы внести залог за многоразовый стакан при покупке напитка?

 Копировать

158 ответов




Вернули бы вы многоразовый стакан после использования для получения залога обратно?

 Копировать

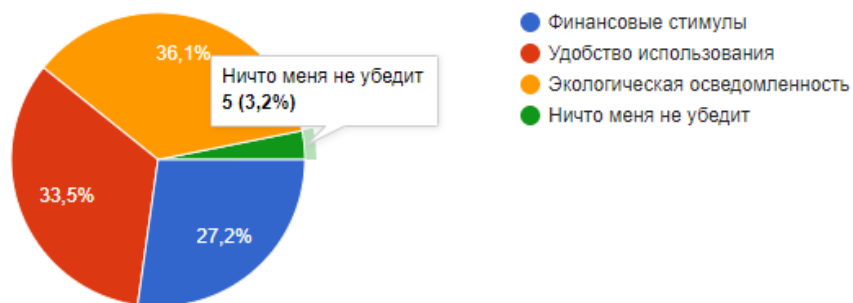
158 ответов



Что могло бы убедить вас начать использовать многоразовые стаканы?

 Копировать

158 ответов



Важно ли для вас наличие точек для возврата многоразовых стаканов рядом с домом или работой?

 Копировать

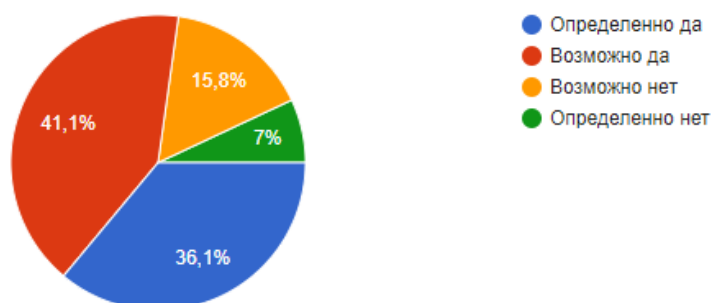
158 ответов




Считаете ли вы, что использование многоразовых стаканов может стать популярным среди большинства людей?

 Копировать

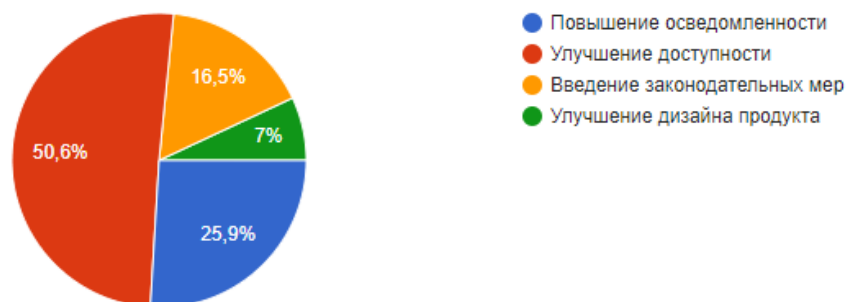
158 ответов




Как вы думаете, какие факторы могут способствовать распространению использования многоразовых стаканов?

 Копировать

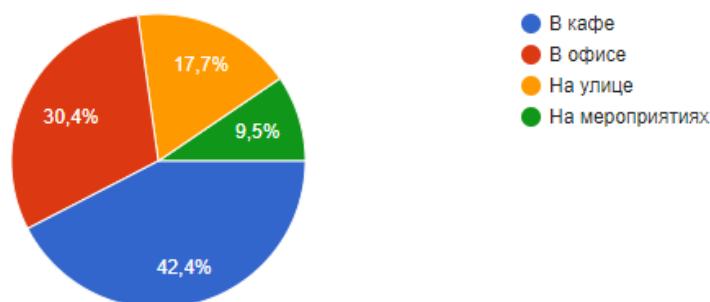
158 ответов



В каких местах вы бы предпочли использовать многоразовые стаканы (кафе, офис, уличные мероприятия)?

 Копировать

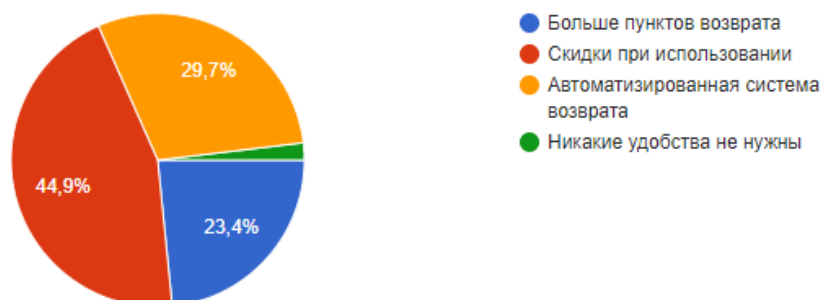
158 ответов




Какие дополнительные удобства или услуги могли бы улучшить ваш опыт использования многоразовых стаканов?

 Копировать

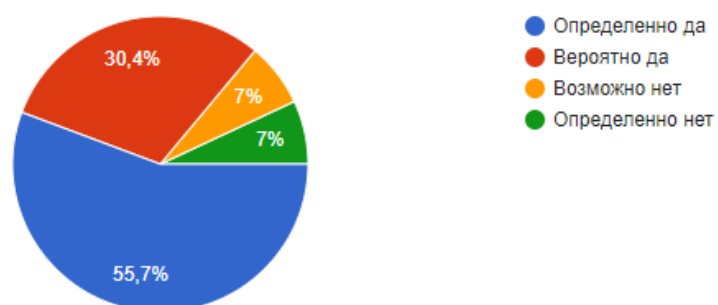
158 ответов



Будете ли вы чувствовать себя лучше, зная, что ваш выбор напитка способствует защите окружающей среды?

 Копировать

158 ответов



Приложение Б
Установка параметров запуска приложения
(справочное)

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { DocumentBuilder, SwaggerModule } from '@nestjs/swagger';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  app.enableCors({
    origin: ['http://localhost:4200'],
    credentials: true
  })

  const config = new DocumentBuilder()
    .setTitle('ShareCUP')
    .setDescription('Бекенд сервиса по капшерингу')
    .setVersion('1.0')
    .build();

  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('/api', app, document);
  app.enableCors();

  await app.listen(3000);
}
bootstrap();
```


Приложение В

Модуль взаимодействия с базой данных

(справочное)

```
import { Module } from '@nestjs/common';
import { AuthModule } from './auth/auth.module';
import { SubscriptionModule } from './subscription/subscription.module';
import { UserModule } from './user/user.module';
import { MongooseModule } from '@nestjs/mongoose';
import { PrivateKeyModule } from './private-key/private-key.module';

@Module({
  imports: [
    MongooseModule.forRoot(
      `mongodb+srv://user:vQTkYnPSRVikD0GJ@cluster0.spmviz0.mongodb.net/dbname?retryWrites=true&w=majority`,
      { dbName: 'cupsharing' },
    ),
    AuthModule,
    SubscriptionModule,
    UserModule,
    PrivateKeyModule
  ],
  controllers: [],
  providers: [],
})
export class AppModule {}
```

Приложение Г
Сервис аутентификации
(справочное)

```
import { Injectable, Logger } from "@nestjs/common";
import { JwtService } from "@nestjs/jwt";
import { Response } from "express";
import { RegisterDto } from "../dto/register.dto";
import { LoginDto } from "../dto/login.dto";
import { UserService } from "../user/user.service";
import * as bcrypt from 'bcryptjs';
import { UserDocument } from "../user/models/user.schema";
import { JWTDecode } from "../shared/interfaces/JWTDecode";

@Injectable()
export class AuthService {

  constructor(
    private jwt: JwtService,
    private user: UserService
  ) {}

  async registerUser(res: Response, dto: RegisterDto) {
    Logger.log(dto);
    if (!dto || !dto.firstName || !dto.lastName || !dto.email || !dto.password ||
    dto.company === undefined) {
      return res.status(400).json({message: 'Проверьте введенные данные'})
    }
  }
}
```

```

const findUser = await this.user.getByEmail(dto.email);
if (findUser)
    return res.status(403).json({message: 'Email уже существует в системе'})

const hashedPassword = await bcrypt.hash(dto.password, 10);

const user = await this.user.createUser({
    ...dto,
    isAdmin: false,
    password: hashedPassword,
    companyInn: dto.companyInn ? dto.companyInn : null,
    companyOgrn: dto.companyOgrn ? dto.companyOgrn : null,
    companyKpp: dto.companyKpp ? dto.companyKpp : null,
    companyAddress: dto.companyAddress ? dto.companyAddress : null,
    companyName: dto.companyName ? dto.companyName : null
});

const token = await this.generateToken(user)

return res.status(201).json(token)

}

async loginUser(res: Response, dto: LoginDto) {
    if (!dto || !dto.email || !dto.password) {
        return res.status(400).json({message: 'Проверьте введенные данные'})
    }

    const findUser = await this.user.getByEmail(dto.email);

```

```
    if (!findUser)
        return res.status(404).json({message: 'Пользователь не найден'})

    const passwordsEqual = await bcrypt.compare(dto.password,
findUser.password);
    if (!passwordsEqual)
        return res.status(404).json({message: 'Неправильный пароль'})

    const token = await this.generateToken(findUser)
    return res.status(201).json(token)
}

async generateToken(user: UserDocument) {
    const tokenPayload: JWTDecode = {
        email: user.email,
        firstName: user.firstName,
        lastName: user.lastName,
    }
    return this.jwt.signAsync(tokenPayload)
}
}
```

Приложение Д
Сервис приватного ключа
(справочное)

```
import { Injectable } from '@nestjs/common';
import { Response } from "express";
import { InjectModel } from "@nestjs/mongoose";
import { PrivateKey, PrivateKeyModel } from "../schemas/private-key.schema";
import { UserService } from "../user/user.service";
import { JwtService } from "@nestjs/jwt";
import { JWTDecode } from "../shared/interfaces/JWTDecode";

@Injectable()
export class PrivateKeyService {

  constructor(
    @InjectModel(PrivateKey.name) private _privateKeyModel: PrivateKeyModel,
    private user: UserService,
    private jwt: JwtService
  ) {} // Используем интерфейс PrivateKeyModel

  // Генерация случайного 6-значного числа
  private generateRandomKey(): string {
    return Math.floor(100000 + Math.random() * 900000).toString();
  }

  // Для запросов физика
  async createNewPrivateKey(res: Response, token: string) {
```

```

const parsedToken: JWTDecode = this.jwt.decode(token)
if (!parsedToken)
  return res.status(404).json({ message: 'Пользователь не найден' })
const user = await this.user.getByEmail(parsedToken.email)
if (!user)
  return res.status(404).json({ message: 'Пользователь не найден' })

// Удалить предыдущие документы
await this._privateKeyModel.deleteByUser(user._id.toString());

// Генерация уникального ключа
let unique = false;
let newKey: string;
while (!unique) {
  newKey = this.generateRandomKey();
  const existingKey = await this._privateKeyModel.findOne({ key: newKey });
  if (!existingKey) {
    unique = true;
  }
}

// Создание нового ключа
const newPrivateKey = new this._privateKeyModel({ user: user._id.toString(),
key: newKey });
await newPrivateKey.save();

return res.status(201).json({ key: newKey, createdAt: newPrivateKey.createdAt,
lifeTimeSeconds: '600' });
}

```

```

async getCurrentPrivateKey(res: Response, token: string) {
  const parsedToken: JWTDecode = this.jwt.decode(token)
  if (!parsedToken)
    return res.status(404).json({ message: 'Пользователь не найден' })
  const user = await this.user.getByEmail(parsedToken.email)
  if (!user)
    return res.status(404).json({ message: 'Пользователь не найден' })

  const keyDocument = await this._privateKeyModel.findOne({ user: user._id })
  if (!keyDocument)
    return res.status(404).json({ message: 'Действующий ключ не найден' })

  return res.status(200).json({
    key: keyDocument.key,
    createdAt: keyDocument.createdAt,
    lifeTimeSeconds: '600'
  })
}

// Экспортируемые функции

async removePrivateKey(key: string) {
  return this._privateKeyModel.findOneAndDelete({ key: key });
}

async getPrivateKey(key: string) {
  return this._privateKeyModel.findOne({ key: key })
}

```

```
}
```

Приложение Е
Сервис подписок
(справочное)

```
import { Injectable } from '@nestjs/common';
import { InjectModel } from '@nestjs/mongoose';
import { Subscription } from 'rxjs';
import { Model } from 'mongoose';
import { SubscriptionDocument } from './models/subscription.model';
import { CreateSubscriptionDto } from './dto/create-subscription.dto';
import { Cups, CupsDocument } from './models/cups.model';
import { Response } from 'express';
import { JwtService } from '@nestjs/jwt';
import { UserService } from '../user/user.service';
import { JWTDecode } from '../shared/interfaces/JWTDecode';
import { InteractionClientCupsDto } from './dto/interaction-client-cups.dto';
import { PrivateKeyService } from '../private-key/private-key.service';
import { InteractionCompanyCupsDto } from './dto/interaction-company-cups.dto';

@Injectable()
export class SubscriptionService {

  constructor(
    @InjectModel(Subscription.name) private subscriptionModel:
    Model<SubscriptionDocument>,
    @InjectModel(Cups.name) private cupsModel: Model<CupsDocument>,
    private jwt: JwtService,
    private user: UserService,
```



```

    private privateKeys: PrivateKeyService
  ) {}

  async currentSubscription(token: string, res: Response) {
    const parsedToken: JWTDecode = this.jwt.decode(token)
    if (!parsedToken)
      return res.status(404).json({ message: 'Пользователь не найден' })
    const user = await this.user.getByEmail(parsedToken.email)
    if (!user)
      return res.status(404).json({ message: 'Пользователь не найден' })

    const userId = user._id as string;

    const subscription = await this.subscriptionModel.findOne({user: userId})

    return res.status(200).json(subscription)
  }

  async getCompanyStatistic(token: string, res: Response) {
    const parsedToken: JWTDecode = this.jwt.decode(token)
    if (!parsedToken)
      return res.status(404).json({ message: 'Пользователь не найден' })
    const barista = await this.user.getByEmail(parsedToken.email)
    if (!barista)
      return res.status(404).json({ message: 'Пользователь не найден' })

    const baristaId = barista._id as string;
    const cups = await this.cupsModel.find({user: baristaId})

```

```

    return res.status(200).json(cups)
  }

  async cupsPrivateTurnover(token: string, res: Response, dto:
InteractionClientCupsDto) {
    const parsedToken: JWTDecode = this.jwt.decode(token)
    if (!parsedToken)
      return res.status(404).json({ message: 'Пользователь не найден' })
    const barista = await this.user.getByEmail(parsedToken.email)
    if (!barista)
      return res.status(404).json({ message: 'Пользователь не найден' })

    const baristaId = barista._id as string;
    switch (dto.type) {
      case "order": return this.incrementCups(baristaId, dto.key, res)
      case "return": return this.decrementCups(baristaId, dto.key, res)
    }
  }

  async cupsCompanyTurnover(token: string, res: Response, dto:
InteractionCompanyCupsDto) {
    const parsedToken: JWTDecode = this.jwt.decode(token)
    if (!parsedToken)
      return res.status(404).json({ message: 'Пользователь не найден' })
    const barista = await this.user.getByEmail(parsedToken.email)
    if (!barista)
      return res.status(404).json({ message: 'Пользователь не найден' })

    const baristaId = barista._id as string;
    switch (dto.type) {

```

```

    case "order": return this.orderCups(baristaId, dto.amount, res)
    case "return": return this.returnCups(baristaId, dto.amount, res)
  }
}

async create(createSubscriptionDto: CreateSubscriptionDto, res: Response, token:
string) {
  const parsedToken: JWTDecode = this.jwt.decode(token)
  if (!parsedToken)
    return res.status(404).json({ message: 'Пользователь не найден' })
  const user = await this.user.getByEmail(parsedToken.email)
  if (!user)
    return res.status(404).json({ message: 'Пользователь не найден' })

  const userId = user._id as string;

  const currentSubscription = await this.subscriptionModel.findOne({ user:
userId })

  if (currentSubscription) {
    const cupsCurrent = currentSubscription.cupsCurrent;
    if (cupsCurrent > this.getAvailableCupsForTypes(createSubscriptionDto.type))
    {
      return res.status(400).json({ message: 'У вас больше взятых кружек, чем
доступно в новой подписке' })
    }
  }
  const newSubscription = await
this.subscriptionModel.findOneAndUpdate({ user: userId }, {
    period: createSubscriptionDto.period,
    type: createSubscriptionDto.type,

```

```

        cupsAvailable: this.getAvailableCupsForTypes(createSubscriptionDto.type),
    }, {new: true}))
    return res.status(201).json(newSubscription)
}

const newSubscription = await this.subscriptionModel.create({
    period: createSubscriptionDto.period,
    type: createSubscriptionDto.type,
    user: userId,
    cupsAvailable: this.getAvailableCupsForTypes(createSubscriptionDto.type),
    cupsCurrent: 0
})

return res.status(201).json(newSubscription)
}

getAvailableCupsForTypes(type: 'privateBase' | 'privateStudent' | 'companyBase' |
'companyPremium') {
    switch (type) {
        case "companyBase": return 600 * 31;
        case "companyPremium": return 1000 * 31;
        case "privateStudent": return 3;
        case "privateBase": return 5;
    }
}

async orderCups(baristaId: string, amount: number, res: Response) {

```

```

    const subscriptionBarista = await this.subscriptionModel.findOne({ user:
baristaId })

    if (!subscriptionBarista) {
        return res.status(404).json({ message: 'Подписка не найдена'})
    }

    if (subscriptionBarista.cupsCurrent + amount >
subscriptionBarista.cupsAvailable)
        return res.status(404).json({ message: 'Вы не можете заказать больше кружек,
чем по подписке'})

    subscriptionBarista.cupsCurrent += amount;
    await subscriptionBarista.save();

    const order = await this.cupsModel.create({
        user: baristaId,
        amount: amount,
        type: 'order',
    })

    return res.status(200).json(order);
}

async returnCups(baristaId: string, amount: number, res: Response) {
    const subscriptionBarista = await this.subscriptionModel.findOne({ user:
baristaId })

    if (!subscriptionBarista) {
        return res.status(404).json({ message: 'Подписка не найдена'})
    }

```

```

    if (subscriptionBarista.cupsCurrent - amount < 0)
        return res.status(404).json({message: 'Вы не можете сдать больше кружек,
чем у вас есть'})

    subscriptionBarista.cupsCurrent -= amount;
    await subscriptionBarista.save();

    const order = await this.cupsModel.create({
        user: baristaId,
        amount: amount,
        type: 'return',
    })

    return res.status(200).json(order);
}

async decrementCups(baristaId: string, privateKey: string, res: Response) {
    const key = await this.privateKeys.getPrivateKey(privateKey)

    if (!key)
        return res.status(404).json({message: 'Ключ не найден, попросите
пользователя сгенерировать новый'})

    const userId = key.user

    await this.privateKeys.removePrivateKey(privateKey)

    const subscriptionBarista = await this.subscriptionModel.findOne({ user:
baristaId })

    const subscriptionUser = await this.subscriptionModel.findOne({ user: userId });

```

```

    if (!subscriptionUser || !subscriptionBarista) {
        return res.status(404).json({ message: 'Подписка не найдена' })
    }
    if (subscriptionUser.cupsCurrent <= 0) {
        return res.status(400).json({ message: 'У пользователя нет взятых кружек' })
    }
    subscriptionBarista.cupsCurrent += 1;
    subscriptionUser.cupsCurrent -= 1;
    await subscriptionBarista.save();
    await subscriptionUser.save();
    return res.status(200).json(null)
}

async incrementCups(baristaId: string, privateKey: string, res: Response) {
    const key = await this.privateKeys.getPrivateKey(privateKey)

    if (!key)
        return res.status(404).json({ message: 'Ключ не найден, попросите
пользователя сгенерировать новый' })

    const userId = key.user;

    await this.privateKeys.removePrivateKey(privateKey)

    const subscriptionUser = await this.subscriptionModel.findOne({ user: userId });
    const subscriptionBarista = await this.subscriptionModel.findOne({ user:
baristaId })

    if (!subscriptionUser || !subscriptionBarista) {

```

```

    return res.status(404).json({ message: 'Подписка не найдена'})
  }
  if (subscriptionUser.cupsCurrent >= subscriptionUser.cupsAvailable) {
    return res.status(400).json({ message: 'У пользователя максимальное
количество кружек по подписке'})
  }
  if (subscriptionBarista.cupsCurrent <= 0) {
    return res.status(400).json({ message: 'У вас не осталось свободных кружек'})
  }

  subscriptionUser.cupsCurrent += 1;
  await subscriptionUser.save();
  return res.status(200).json(null)
}

async getAllStatistic(res: Response, token: string) {
  const parsedToken: JWTDecode = this.jwt.decode(token)
  if (!parsedToken)
    return res.status(404).json({ message: 'Пользователь не найден'})
  const user = await this.user.getByEmail(parsedToken.email)
  if (!user)
    return res.status(404).json({ message: 'Пользователь не найден'})
  if (!user.isAdmin)
    return res.status(403).json({ message: 'Пользователь не админ'})

  const cups = await this.cupsModel.find().populate('user').exec()

  return res.status(200).json(cups)
}

```


}

Приложение Ж
Сервис пользователя
(справочное)

```
import { Injectable } from "@nestjs/common";
import { JwtService } from "@nestjs/jwt";
import { JWTDecode } from "../shared/interfaces/JWTDecode";
import { Response } from "express";
import { Model } from "mongoose";
import { User, UserDocument } from "../models/user.schema";
import { InjectModel } from "@nestjs/mongoose";
import { RegisterDto } from "../auth/dto/register.dto";
import * as bcrypt from 'bcryptjs';

@Injectable()
export class UserService {

  constructor(
    private jwt: JwtService,
    @InjectModel(User.name) private _userModel: Model<UserDocument>,
  ) {
  }

  async getTokenizedUser(res: Response, token: string) {
    const parsedToken: JWTDecode = this.jwt.decode(token)
    if (!parsedToken)
      return res.status(404).json({ message: 'Пользователь не найден' })
    const user = await this._userModel.findOne({ email: parsedToken.email })
    if (!user)
      return res.status(404).json({ message: 'Пользователь не найден' })
  }
}
```

```

return res.status(200).json(user)
}

async editTokenizedUser(res: Response, body: RegisterDto, token: string) {
  const parsedToken: JWTDecode = this.jwt.decode(token)
  if (!parsedToken)
    return res.status(404).json({ message: 'Пользователь не найден' })
  const user = await this._userModel.findOne({ email: parsedToken.email })
  if (!user)
    return res.status(404).json({ message: 'Пользователь не найден' })

  let editedUser: any = {
    email: body.email,
    firstName: body.firstName,
    lastName: body.lastName,
  }

  if (user.company) {
    editedUser = {
      ...editedUser,
      companyInn: body.companyInn ? body.companyInn : null,
      companyOgrn: body.companyOgrn ? body.companyOgrn : null,
      companyKpp: body.companyKpp ? body.companyKpp : null,
      companyAddress: body.companyAddress ? body.companyAddress : null,
      companyName: body.companyName ? body.companyName : null
    }
  }

  if (body.password) {
    const hashedPassword = await bcrypt.hash(body.password, 10);

```

```

    editedUser = {
      ...editedUser,
      password: hashedPassword
    }
  }

  const updatedUser = await this._userModel.findOneAndUpdate({email:
parsedToken.email}, editedUser, {new: true})
  return res.status(200).json(updatedUser)
}

async getUsers() {
  return this._userModel.find()
}

async getUser(id: string) {
  return this._userModel.findById(id)
}

async getByEmail(email: string) {
  return this._userModel.findOne({email: email})
}

async createUser(dto: User) {
  return this._userModel.create(dto);
}
}

```

Приложение 3
Компонент логин
(справочное)

```
import { Component, OnDestroy } from '@angular/core';
import { FormControl, FormGroup, ReactiveFormsModule, Validators } from
"@angular/forms";
import { TuiInputModule, TuiInputPasswordModule } from "@taiga-ui/kit";
import { AuthorizationService } from "../../authorization.service";
import { take, takeWhile } from "rxjs";
import { CookieService } from "ngx-cookie-service";
import { Router } from "@angular/router";
import { TuiButtonModule } from "@taiga-ui/core";

@Component({
  selector: 'app-login',
  standalone: true,
  imports: [
    ReactiveFormsModule,
    TuiInputModule,
    TuiInputPasswordModule,
    TuiButtonModule
  ],
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnDestroy {
  loginForm!: FormGroup;

  alive = true;
```

```
error: string | null = null;
```

```
loading = false;
```

```
constructor(private auth: AuthorizationService, private cookie: CookieService,  
private router: Router) {
```

```
  this.loginForm = new FormGroup<any>({  
    email: new FormControl(null),  
    password: new FormControl(null)  
  })
```

```
  this.loginForm  
    .valueChanges  
    .pipe(takeWhile(() => this.alive))  
    .subscribe(() => {  
      this.error = null  
    })
```

```
}
```

```
login() {  
  this.error = null;  
  this.loading = true;  
  this.auth.loginUser(this.loginForm.value)  
    .pipe(take(1))  
    .subscribe({  
      next: (value) => {  
        const token = value as string;  
        this.cookie.set('authToken', token, this.add24HoursAndOutputJSON())
```

```
        this.router.navigateByUrl('cabinet')
    },
    error: err => {
        this.error = err.error.message;
        this.loading = false;
    }
})
}

ngOnDestroy() {
    this.alive = false;
}

add24HoursAndOutputJSON() {
    const currentDate = new Date();
    currentDate.setHours(currentDate.getHours() + 24);
    return currentDate
}

}
```

Приложение И
Компонент регистрации
(справочное)

```
import { Component, OnDestroy } from '@angular/core';
import { FormControl, FormGroup, ReactiveFormsModule, Validators } from
"@angular/forms";
import { TuiInputModule, TuiInputPasswordModule, TuiToggleModule } from
"@taiga-ui/kit";
import { AuthorizationService } from "../../authorization.service";
import { CookieService } from "ngx-cookie-service";
import { Router } from "@angular/router";
import { take, takeWhile } from "rxjs";
import { TuiButtonModule } from "@taiga-ui/core";

@Component({
  selector: 'app-register',
  standalone: true,
  imports: [
    ReactiveFormsModule,
    TuiInputModule,
    TuiInputPasswordModule,
    TuiToggleModule,
    TuiButtonModule
  ],
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.scss']
})
export class RegisterComponent implements OnDestroy {
  registerForm!: FormGroup;
```



```
alive = true;
```

```
error: string | null = null;
```

```
loading = false;
```

```
constructor(private auth: AuthorizationService, private cookie: CookieService,  
private router: Router) {
```

```
    this.registerForm = new FormGroup<any>({  
        email: new FormControl(null, Validators.required),  
        password: new FormControl(null, Validators.required),  
        firstName: new FormControl(null, Validators.required),  
        lastName: new FormControl(null, Validators.required),  
        company: new FormControl<boolean>(false, Validators.required),  
        companyInn: new FormControl(null),  
        companyOgrn: new FormControl(null),  
        companyKpp: new FormControl(null),  
        companyAddress: new FormControl(null),  
        companyName: new FormControl(null),  
    })
```

```
    this.registerForm.valueChanges.subscribe(() => {  
        this.error = null;  
    })
```

```
    this.registerForm.controls['company'].valueChanges  
        .pipe(takeWhile(() => this.alive))  
        .subscribe(state => {  
            if (state === true) {
```

```
this.registerForm.controls['companyInn'].addValidators([Validators.required])

this.registerForm.controls['companyOgrn'].addValidators([Validators.required])

this.registerForm.controls['companyKpp'].addValidators([Validators.required])

this.registerForm.controls['companyAddress'].addValidators([Validators.required])

this.registerForm.controls['companyName'].addValidators([Validators.required])
    } else {
        this.registerForm.patchValue({
            companyInn: null,
            companyOgrn: null,
            companyKpp: null,
            companyAddress: null,
            companyName: null,
        })

this.registerForm.controls['companyInn'].removeValidators([Validators.required])

this.registerForm.controls['companyOgrn'].removeValidators([Validators.required]
)

this.registerForm.controls['companyKpp'].removeValidators([Validators.required])

this.registerForm.controls['companyAddress'].removeValidators([Validators.required])
```

```

this.registerForm.controls['companyName'].removeValidators([Validators.required
])
    }
    this.registerForm.controls['companyInn'].updateValueAndValidity();
    this.registerForm.controls['companyOgrn'].updateValueAndValidity();
    this.registerForm.controls['companyKpp'].updateValueAndValidity();
    this.registerForm.controls['companyAddress'].updateValueAndValidity();
    this.registerForm.controls['companyName'].updateValueAndValidity();
    })
}

register() {
    this.error = null;
    this.loading = true;
    this.auth.registerUser(this.registerForm.value)
        .pipe(take(1))
        .subscribe({
            next: (value) => {
                const token = value as string;
                this.cookie.set('authToken', token, this.add24HoursAndOutputJSON())
                this.router.navigateByUrl('cabinet')
            },
            error: err => {
                this.error = err.error.message;
                this.loading = false;
            }
        })
}

```

```
ngOnDestroy() {  
  this.alive = false;  
}  
  
add24HoursAndOutputJSON() {  
  const currentDate = new Date();  
  currentDate.setHours(currentDate.getHours() + 24);  
  return currentDate  
}  
}
```