1. **Тема для разработки веб-сайта**

Приложение для поиска местных фермеров и рынков.

1. **Предметная область**

*Заказчик:*

* Союз фермеров Ленинградской области и Санкт-Петербурга.

*Сроки:*

* Ввод в эксплуатацию первичной версии продукта в неполном функционале в течение 1 года с момента начала разработки.
* В течение 3 лет с момента ввода в эксплуатацию планируются регулярные обновления на основе отзывов заказчика и пользователей, а также внедрение роботизированной доставки.

*Основные функции:*

* Поиск по геолокации.
* Каталог фермеров и рынков
* Отзывы и рейтинги
* Календарь событий
* Система уведомлений
* Рецепты и советы
* Сообщество
* Опция предварительного заказа
* Интеграция с социальными сетями
* Образовательный контент

*Уровни доступа к сайту:*

* Фермеры: регистрация, создание профиля, добавление продукции, управление заказами и взаимодействие с покупателями.
* Покупатели: поиск фермеров и рынков, просмотр предложений, оформление заказов, оставление отзывов.
* Администраторы: управление контентом, модерация пользователей, анализ данных и поддержка пользователей.
* Логисты: координация доставки, управление маршрутами и взаимодействие с фермерами и покупателями.
* Робототехники: внедрение роботизированной доставки.

*Особенности клиента:*

* Структура: физические лица, ИП и малые/средние фермерские хозяйства.
* Местоположение: приложение ориентировано на городские и пригородные районы, где есть спрос на местные продукты.
* Прочие особенности: учитывать предпочтения пользователей в отношении органических продуктов, устойчивого развития и поддержки местных производителей.

*Сложность функционала ПО:*

* Базовые функции: регистрация и авторизация пользователей, поиск фермеров и рынков по геолокации, просмотр продукции.
* Продвинутые функции: интеграция с платёжными системами, система отзывов и рейтингов, аналитика покупок, возможность подписки на новости от фермеров, функция уведомлений о новых поступлениях.

*Требования к надёжности:*

* Надёжность системы: минимум 99% времени безотказной работы, быстрый отклик (не более 2 секунд на запрос), регулярное резервное копирование данных и защита личной информации пользователей.
* Первоначальное количество личных кабинетов: 100 000. С возможностью дальнейшего бесперебойного расширения.
* Безопасность: защита от несанкционированного доступа и утечек персональных и платежных данных.

*Технологический стек:*

* Frontend: React или Vue.js для веб-приложения; React Native или Flutter для мобильного приложения.
* Backend: Node.js с Express или Python с Django для серверной части.
* База данных: MongoDB или PostgreSQL для хранения информации о пользователях, продуктах и рынках.
* API: использование картографических API (например, Google Maps) для отображения местоположений.
* Роботизированная система доставки: на основе опыта Яндекс Доставки в Москве и Иннополисе.

*Монетизация:*

* Финансирование заказчика.
* Реклама от местных производителей.
* Премиум-подписка для фермеров с дополнительными функциями (например, продвинутые аналитические инструменты).
* Партнёрство с магазинами здорового питания или ресторанами.

*Сильные и слабые стороны предметной области:*

*Сильные стороны:*

1. **Поддержка местной экономики**: Приложение способствует развитию местных производителей и фермеров, что может привести к экономическому росту региона и созданию новых рабочих мест.

**2. Экологичность**: Уменьшение углеродного следа за счет сокращения расстояния, на которое продукты транспортируются, способствует защите окружающей среды и снижению негативного воздействия на климат.  
  
3. **Свежесть продуктов**: Пользователи имеют возможность получать свежие и качественные продукты непосредственно от фермеров, что может повысить уровень удовлетворенности потребителей.  
  
4. **Образование**: Приложение может предоставлять ценную информацию о методах ведения сельского хозяйства, органических продуктах и сезонности, что помогает пользователям делать более осознанный выбор.  
  
5. **Сообщество**: Фостеринг сообщества между потребителями и производителями укрепляет связи на уровне местного населения, что может привести к более устойчивым отношениям и поддержке локальных инициатив.  
  
6. **Удобство поиска**: Упрощение процесса поиска и выбора местных фермеров и рынков с использованием фильтров по расстоянию, типу продукции и другим критериям делает использование приложения более комфортным для пользователей.

*Слабые стороны:*

1. Ограниченная база данных: в некоторых регионах может быть недостаточно фермеров или рынков, что ограничивает выбор для пользователей.
2. Технические проблемы: возможны сложности с доступом к интернету в сельских районах, что может затруднить использование приложения.
3. Отсутствие стандартизации: разные фермеры могут иметь разные методы ведения бизнеса, что может вызывать путаницу у пользователей.
4. Конкуренция с крупными ритейлерами: местные фермеры могут не иметь ресурсов для конкуренции с большими супермаркетами по цене и ассортименту.
5. Проблемы с логистикой: доставка продуктов может быть сложной задачей, особенно в удаленных районах.
6. Сезонность: доступность продуктов может зависеть от сезона, что может ограничивать выбор в определенное время года.
7. **Роли команды разработчиков (14 человек)**

* Ведущий разработчик
* Младший разработчик
* Стажёр-разработчик
* Архитектор
* Системный администратор
* Бизнес-аналитик
* Технический писатель
* IS Control – контроль ИБ
* Релиз-менеджер
* Менеджер проекта
* Ведущий тестировщик
* Младший тестировщик
* Инженер
* Робототехник

1. **Смотреть пункт 2**
2. **Модели жизненного цикла**

1. Водопадная модель

Описание: линейный подход, где каждый этап (анализ, проектирование, реализация, тестирование, внедрение) выполняется последовательно.

Плюсы:

1. Четкая структура: водопадная модель предполагает последовательное выполнение этапов (анализ требований, проектирование, реализация, тестирование и внедрение). Это позволяет команде четко планировать и контролировать процесс.
2. Легкость в управлении: каждый этап завершен перед переходом к следующему, что упрощает управление проектом и позволяет легко отслеживать прогресс.
3. Документация: водопадная модель требует обширной документации на каждом этапе, что может быть полезно для понимания требований и дизайна приложения.
4. Согласование требований: на начальных этапах можно тщательно проработать требования, что может уменьшить вероятность недопонимания между командой разработки и заказчиком.
5. Подходит для небольших проектов: если проект небольшой и требования хорошо определены, водопадная модель может быть эффективной.

Минусы:

1. Недостаточная гибкость: изменения в требованиях на поздних стадиях могут быть сложными и затратными, что может привести к проблемам, если рынок или потребности пользователей изменятся.
2. Риски на этапе тестирования: тестирование происходит только после завершения разработки, что может привести к обнаружению серьезных проблем на поздних стадиях.
3. Долгий цикл разработки: водопадная модель не подходит для проектов с неопределенными требованиями или, когда нужно быстро реагировать на изменения.
4. Отсутствие итераций: нет возможности для итеративного улучшения продукта на основе обратной связи от пользователей в процессе разработки.
5. Может не учитывать потребности пользователей: если требования были неправильно поняты на этапе анализа, это может привести к созданию приложения, которое не соответствует ожиданиям пользователей.

2. Гибкая модель разработки

Описание: итеративный подход, который предполагает частые изменения и вовлечение пользователей на всех этапах разработки.

Плюсы:

1. Гибкость и адаптивность: позволяет быстро реагировать на изменения в требованиях и потребностях пользователей, что особенно важно в условиях динамичного рынка.
2. Итеративный процесс: проект разбивается на небольшие циклы (спринты), что позволяет регулярно выпускать обновления и получать обратную связь от пользователей.
3. Участие заинтересованных сторон: заказчики и пользователи могут активно участвовать в процессе разработки, что помогает лучше понять их потребности и ожидания.
4. Улучшение качества продукта: регулярное тестирование и демонстрации промежуточных результатов способствуют выявлению и устранению проблем на ранних стадиях.
5. Повышение мотивации команды: способствует более высокому уровню вовлеченности и ответственности команды, так как каждый член понимает свою роль в достижении общей цели.
6. Фокус на ценности для пользователя: гибкая модель разработки ориентирован на создание функционала, который приносит реальную ценность пользователям, что важно для успешного приложения.

Минусы:

1. Неопределенность в планировании: из-за гибкости могут возникнуть трудности с долгосрочным планированием, что может привести к недоразумениям в сроках и бюджете.
2. Требует высокой вовлеченности: гибкая модель требует активного участия всех заинтересованных сторон, что может быть сложно обеспечить, особенно если у заказчика нет достаточного времени или ресурсов.
3. Сложности с документацией: в гибкой модели меньше акцента на документацию, что может привести к недостаточной информации о проекте для новых участников команды или при передаче проекта.
4. Риск «бесконечного» процесса: из-за итеративного подхода может возникнуть ситуация, когда проект никогда не будет завершен, так как всегда будут новые идеи и улучшения.
5. Необходимость сильной команды: гибкая модель требует от команды высокой степени самоорганизации и ответственности, что может быть проблемой для менее опытных групп.

3. Модель V-образного цикла

Описание: модель, которая сочетает в себе элементы водопадной модели и тестирования, где каждый этап проектирования соответствует этапу тестирования.

Плюсы:

1. Четкая структура: модель V-образного цикла имеет четко определенные этапы, что облегчает планирование и управление проектом.

2. Раннее выявление ошибок: тестирование начинается на ранних этапах разработки, что позволяет выявлять и исправлять ошибки до выхода на финальные стадии.

3. Документация: модель требует наличия документации на каждом этапе, что помогает сохранить знания о проекте и облегчает его поддержку в будущем.

4. Легкость в управлении: четкие этапы и взаимосвязи между ними упрощают управление проектом и позволяют легко отслеживать прогресс.

5. Подходит для стабильных требований: если требования к приложению хорошо известны и не подлежат частым изменениям, V-Model может быть эффективным выбором.

Минусы:

1. Негибкость: модель V-образного цикла плохо справляется с изменениями требований. Если в процессе разработки возникают новые идеи или изменения, это может привести к значительным затратам времени и ресурсов.

2. Долгий цикл разработки: поскольку все этапы должны быть завершены последовательно, это может увеличить общее время разработки, особенно если возникают задержки на ранних этапах.

3. Трудности с тестированием: хотя тестирование начинается рано, оно все же зависит от завершения предыдущих этапов, что может затруднить обнаружение проблем на более поздних стадиях.

4. Не подходит для сложных проектов: если проект требует частых изменений или имеет неопределенные требования, модель V может оказаться неэффективной.

5. Сложность в управлении рисками: из-за своей линейной структуры V-Model может не учитывать риски, связанные с изменениями в требованиях или технологиях.

**Обоснование выбора модели**

Выбор: Scrum

Обоснование:

1. **Итеративный подход**: В условиях динамично меняющегося рынка, где предпочтения местных фермеров и покупателей могут меняться, итеративный подход Scrum позволяет команде регулярно проверять и адаптировать продукт. Каждая итерация (спринт) предоставляет возможность собирать обратную связь от пользователей, что особенно важно для понимания их потребностей и ожиданий. Это помогает команде своевременно вносить изменения и улучшения, что способствует созданию более релевантного и востребованного продукта.
2. **Прозрачность и вовлеченность**: Включение заказчика и пользователей в процесс разработки через активное участие в формировании бэклога позволяет лучше понять их потребности и ожидания. Это критично для «Cowwow», поскольку успешность приложения зависит от того, насколько хорошо оно отвечает запросам местных фермеров и покупателей. Прозрачность процесса разработки также создает доверие между командой и пользователями, что может привести к более активному участию последних в тестировании и улучшении продукта.
3. **Управление рисками**: Регулярные встречи, такие как ежедневные стендапы и спринт-ревью, позволяют команде быстро выявлять потенциальные проблемы и риски. Например, если команда сталкивается с трудностями в интеграции функций поиска или отображения информации о рынках, это можно обсудить на встречах и найти решение до того, как проблема станет критической. Такой проактивный подход к управлению рисками минимизирует вероятность задержек и повышает качество конечного продукта.
4. **Фокус на ценности**: Scrum помогает команде сосредоточиться на создании функционала, который приносит максимальную ценность пользователям. Для «Cowwow» это может означать приоритетное развитие функций, которые облегчают поиск фермеров или упрощают процесс покупки продуктов. Такой фокус позволяет избежать излишней работы над менее значительными функциями, что особенно важно в условиях ограниченных ресурсов.
5. **Командная работа**: Scrum поощряет сотрудничество внутри команды, что способствует обмену знаниями и улучшению качества продукта. В контексте «Cowwow» это может быть полезно для интеграции различных экспертиз — например, разработчиков, дизайнеров и специалистов по пользовательскому опыту — для создания более интуитивного и функционального приложения. Командная работа также укрепляет дух единства, что важно для достижения общих целей.
6. **Смотреть пункт 5**
7. **Виды деятельности и этапы**

*ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ*

[6.3.1.3](https://6.3.1.3/).1.3 По мере необходимости и при согласии всех заинтересованных сторон требования проекта могут быть изменены на этом этапе для достижения критериев завершения.

6.2.4.3.3.4 Гарантировать обеспечение обратной связи от результатов выполненных оценок к штатному персоналу.

*ПРОЕКТИРОВАНИЕ*

[7.1.3.3](https://7.1.3.3/).1 Проектирование архитектуры программных средств Для каждого программного элемента (или элемента конфигурации, если он определен) данный вид деятельности состоит из решения следующих задач:

[7.1.3.3](https://7.1.3.3/).1.1 Исполнитель должен преобразовать требования к программным составным частям в архи­тектуру, которая описывает верхний уровень его структуры и идентифицирует программные компоненты. Необходимо гарантировать, что все требования к программным составным частям распределяются по про­граммным компонентам и в дальнейшем уточняются для облегчения детального проектирования. Архитек­туру программной составной части необходимо документировать.

[7.1.4.3](https://7.1.4.3/).1.2 Исполнитель должен разработать и документально оформить детальный проект для вне­шних интерфейсов к программным составным частям, между программными компонентами и между про­граммными блоками. Необходимо, чтобы детальный проект для интерфейсов позволял проводить кодиро­вание без потребности в получении дополнительной информации. [7.1.4.3](https://7.1.4.3/).1.3 Исполнитель должен разработать и документально оформить детальный проект базы данных.

*РЕАЛИЗАЦИЯ*

[7.1.5.3](https://7.1.5.3/) Виды деятельности и задачи При реализации проекта необходимо выполнять следующие виды деятельности и задачи в соответствии с принятыми в организации политиками и процедурами в отношении процесса конструирования программных средств.

*ТЕСТИРОВАНИЕ*

[7.1.5.3](https://7.1.5.3/).1.2 Исполнитель должен тестировать каждый программный блок и базу данных, гарантируя, что они удовлетворяют требованиям. Результаты тестирования должны быть документально оформлены.

*ВНЕДРЕНИЕ*

[6.3.1.3](https://6.3.1.3/).2.1 Планы должны включать в себя, по крайней мере: а) графики работ для своевременного завершения задач; b) оценку усилий; с) ресурсы, необходимые для выполнения задач; d) распределение задач; е) распределение обязанностей; f) количественное определение рисков, связанных с задачами или самим процессом; g) мероприятия по гарантии качества для применения в пределах всего проекта; h) затраты, связанные с выполнением процесса; і) обеспечение окружающей среды и инфраструктуры; j) определение и сопровождение модели жизненного цикла, состоящей из стадий, используя конкретные модели жизненного цикла для проектов организации.

[6.3.1.3](https://6.3.1.3/).3 Активизация проекта Данный вид деятельности состоит из решения следующих задач:

[6.3.1.3](https://6.3.1.3/).3.1 Менеджер должен получить полномочия на проект.

[6.3.1.3](https://6.3.1.3/).3.2 Менеджер должен представить заявки на необходимые ресурсы для выполнения проекта.

[6.3.1.3](https://6.3.1.3/).3.3 Менеджер должен инициировать выполнение планов проекта для удовлетворения совокупности целей и критериев осуществления управления проектом.

*ЭКСПЛУАТАЦИЯ*

[6.4.9.3](https://6.4.9.3/).1.2 Оператор должен определять процедуры для получения, регистрации, решения, просле­живания проблем и обеспечения обратной связи. Всякий раз, когда возникают проблемы, они должны быть зарегистрированы и введены в процесс решения проблем программных средств

[6.2.5.3](https://6.2.5.3/).1.6 Организация должна проводить мониторинг состояния совершенствования продукции и услуг в области качества.

[6.4.9.3](https://6.4.9.3/).1.2 Оператор должен определять процедуры для получения, регистрации, решения, просле­живания проблем и обеспечения обратной связи. Всякий раз, когда возникают проблемы, они должны быть зарегистрированы и введены в процесс решения проблем программных средств.

[6.4.9.3](https://6.4.9.3/).4.1 Оператор должен обеспечивать содействие и консультации пользователей по их просьбе. Эти заявки и последующие действия должны быть зарегистрированы и проконтролированы.

[6.4.9.3](https://6.4.9.3/).4.2 Оператор должен направлять заявки пользователей (по мере необходимости) для выпол­нения в процессе сопровождения программных средств (см. 6.4.10). Все решения должны контролироваться для заключения об их результативности.

[6.4.9.3](https://6.4.9.3/).5.1 Оператор должен направлять возникшие проблемы в процесс решения проблем в про­граммных средствах для их устранения.

*ПОДДЕРЖКА*

[6.2.5.3](https://6.2.5.3/).2 Корректирующие действия менеджмента качества Данный вид деятельности состоит из решения следующих задач:

[6.2.5.3](https://6.2.5.3/).2.1 Организация должна предпринимать корректирующие действия, если конечные цели менеджмента качества не достигнуты.

[6.2.5.3](https://6.2.5.3/).2.2 Организация должна выполнять корректирующие действия и сообщать о результатах в пределах организации.

[6.3.2.3](https://6.3.2.3/).2 Управление проектом Данный вид деятельности состоит из решения следующих задач:

[6.3.2.3](https://6.3.2.3/).2.1 Менеджер должен исследовать, проанализировать и принять решения по проблемам, обнаруженным при выполнении проекта.

[6.3.2.3](https://6.3.2.3/).2.2 Менеджер должен представлять отчеты в согласованные сроки о развитии проекта, показывая соблюдение планов и решения в случае остановки в развитии проекта.

ЗАВЕРШЕНИЕ ПРОЕКТА

[7.2.3.2](https://7.2.3.2/) В результате успешного осуществления процесса гарантии качества программных средств: а) разрабатывается стратегия обеспечения гарантии качества; b) создается и поддерживается свидетельство гарантии качества; с) идентифицируются и регистрируются проблемы и (или) несоответствия с требованиями; d) верифицируется соблюдение продукцией, процессами и действиями соответствующих стандартов, процедур и требований.

УЛУЧШЕНИЕ ПРОЦЕССОВ

[6.3.2.3](https://6.3.2.3/).3 Оценка проекта Данный вид деятельности состоит из решения следующих задач:

[6.3.2.3](https://6.3.2.3/).4 Завершение проекта Данный вид деятельности состоит из решения следующих задач:

[6.3.2.3](https://6.3.2.3/).4.1 Когда все программные продукты, действия и задачи завершаются, менеджер должен определить, закончен ли проект, принимая во внимание критерии, указанные в контракте или установленные как часть процедуры организации.

[6.3.2.3](https://6.3.2.3/).4.2 Эти результаты и отчеты должны быть архивированы в соответствующей среде, как определено в контракте.

1. **Смотреть пункт 7**
2. **Построенная модель жизненного цикла**
3. *Подготовительный этап*

Сбор требований: определение основных потребностей пользователей и функций (поиск фермеров, отображение информации о продуктах и т.д.).

Создание бэклога: формирование списка задач (user stories), которые должны быть выполнены в ходе разработки.

1. *Формирование команды*

Смотреть пункт 3.

1. *Планирование спринта*

Определение длительности спринта: выбрать продолжительность спринта (2-4 недели).

Планирование спринта: провести встречу по планированию спринта, чтобы выбрать задачи из бэклога, которые будут выполнены в текущем спринте.

1. *Разработка*

Итеративная разработка: разработка функционала по частям, каждая из которых должна быть завершена к концу спринта.

1. *Тестирование*

Тестирование в процессе разработки: проводить тестирование новых функций параллельно с их разработкой, чтобы выявлять и исправлять ошибки на ранних стадиях.

Интеграционное тестирование: убеждаться, что новые функции корректно работают вместе с уже существующими.

1. *Демонстрация*

Демонстрация результатов: в конце спринта провести демонстрацию (sprint review) для заинтересованных сторон, показать, что было сделано.

Сбор обратной связи: получить отзывы от пользователей и заинтересованных сторон о новой функциональности.

1. *Ретроспектива*

Проведение ретроспективы: после демонстрации провести встречу для обсуждения того, что прошло хорошо, что можно улучшить и какие действия необходимо предпринять для улучшения процессов в следующем спринте.

1. *Обновление бэклога*

Анализ и обновление бэклога: на основе полученной обратной связи обновить бэклог, добавив новые задачи и приоритизировав их.

1. *Повторение процесса*

Начало нового спринта: повторять шаги 3-8, добавляя новые функции и улучшая приложение на основе обратной связи.

1. *Завершение проекта*

Оценка успешности проекта: провести финальный обзор проекта, оценив выполнение целей и удовлетворенность пользователей.

Подготовка итогового отчета: создать документ с результатами работы над проектом.

Закрытие проекта: формально завершить проект, включая архивирование документации и материалов.