Национальный технический университет Украины
«Киевский политехнический институт»
Факультет информатики и вычислительной техники
Кафедра вычислительной техники

Проект

на тему : «Система реализации связи заказчик-разработчик типа фриланс»

Виполнил:
Студент группы ИВ-32
Довгаль Д.С.
Зачетная книжка №3213
Проверил: Болдак А. А.

г. Киев
2014 р.

Запросы заинтересованных лиц

1. Введение

В данном документе описываются запросы заинтересованных по отношению к разрабатываемой системе «Система реализации связи заказчик-разработчик типа «фриланс», в качестве которых выступают: заказчик – любая частная фирма, желающая реализовать данную систему, работодатели, исполнители заказов-«фрилансеры».

1.1. Цель

Целью документа является определить главные требования к функциональности, производительности, надежности, удобства, доступности, а также определить бизнес правила и технологические ограничения, накладываемые на предмет разработки.

1.2. Контекст

Перечень требований, указанных в данном документе, являются основой технического задания для разработки данной системы связи заказчик-разработчик.

2. Краткий обзор продукта

Система реализации связи заказчик-разработчик типа «фриланс» - это база данных работодателей и исполнителей заказов-«фрилансеров», а также руководство сайта. Система содержит 2 различных типа доступа:

- Заказчики могут размещать заказы на выполнение работы в определенной сфере деятельности, привлекая при этом к их выполнению исполнителей. Для становления заказчиком необходима регистрация на сайте и внесение всей необходимой информации для возможности дальнейшей связи со стороны исполнителей или администрации сайта. При возникновении определенных проблем заказчик может обратиться к администрации для последующего решения возникшей проблемы, а также, в случае конфликта с определенным исполнителем, подать на него жалобу.

- Исполнители «фрилансеры» ищут приемлемый для себя заказ, размещенный заказчиком, договариваются с заказчиком о сроках, условиях выполнения работы, а также о стоимости, выполняют заказ, получают определенную плату за проделанную работу. Аналогично заказчикам, при возникновении определенных проблем могут обращаться к администрации, подавать жалобу на заказчика.

Окончанием сделки заказчика и разработчика является:

- выполненное задание со стороны разработчика (с выполнением всех требований заказчика);

- оговоренная плата разработчику со стороны заказчика (если такая имеется)

3. Деловые правила

3.1. Назначение системы

Система предназначена для удобной удаленной связи заказчика и разработчика, контроля выполнения заказа заказчиком. Система способна избавить данные два заинтересованных лица от прямого контакта и нацелена на обеспечение обычной работы между заинтересованными лицами на большем расстоянии.

3.2. Политика взаимоотношений с клиентом

Клиентами системы могут быть заказчики и разработчики, зарегистрированные в данной системе. Политика взаимоотношений с клиентом системы заключается предоставлении ему разного рода информации:

- заказчикам – список разработчиков, их уровень квалификации, портфолио, отзывы других заказчиках о качестве выполнения заказа данным разработчиком.

- разработчикам – список заказов, полную информацию о требованиях данного проекта, оплате, сроках сдачи. Список заказчиков, их рейтинг, составленный на основе отзывов разработчиков, принимавших у них заказ.

3.3. Характеристика делового процесса

После регистрации заказчиков и разработчиков, которая проводиться согласно определенному сценарию, зарегистрированные пользователи могут заполнять свою страничку пользователя на сайте, а также выкладывать примеры своих работ (для разработчиков) и предлагаемые проекты (для заказчиков), с полным описанием характеристик. Таких как: сроки выполнения, предполагаемый заработок, и пожелания по поводу выполнения.

На сайте есть рейтинги новых проектов. Рейтинг разработчиков составляется опираясь на оценки лиц, которые сотрудничали с ними, аналогично для рейтинга разработчиков. Администраторы контролируют, чтобы выполненные проекты удалялись из списка новых, а список регулярно обновлялся.

Пользователи сайта имеют возможность отправить заявку з жалобой на другого пользователя администратору. При этом в заявке должна быть подробно описана конфликтная ситуация, чтобы администратор мог принять решение.

На страничке разработчика расположена графа, которая говорит о том занят ли он на каком-нибудь проекте и стоит ли его беспокоить разработчику.

Также и с проектом находиться графа, определяющая количество кандидатов на проект.

3.4. Сценарий регистрации нового пользователя

Пользователь регистрируется самостоятельно, заполняя регистрационную форму на сайте, вводит все необходимые данные. Для разработчиков и для заказчиков два разных вида анкет, которые отличаются вводимыми данными.

Общие данные: логин, пароль, подтверждение пароля, ФИО, дата рождения, пол. Отдельно для разработчика: опыт работы, образование, список заказчиков либо компаний, с которыми сотрудничал. Для заказчика дополнительно вносится принадлежность к определенной компании, если она есть. Дополнительные данные вводятся по желанию пользователя.

3.5. Сценарии различных услуг системы

3.5.1. Разработчик – Заказчик

Разработчик выкладывает план проектной работы с необходимыми характеристиками, после чего может выбрать одного или нескольких разработчиков на проект. В то же время разработчик может сам подбирать себе проекты для выполнения и предлагать свою кандидатуру заказчику. В последствии, если проект выполнен благополучно, заказчик выплачивает оговоренную заранее суму за работу.

3.5.2. Администратор – Заказчик и Разработчик

Администратор принимает жалобы от пользователей сайта, рассматривает их, в случае нарушения этических правил, удаляет учетную запись нарушителя.

4. Функциональность

Основные требования к функциональности, предъявленные заинтересованным особам, относятся к 3 категориям:

1. Заказчик
2. Исполнитель «фрилансер»
3. Администратор

4.1 Структура электронная заявка Заказчика

Электронная заявка должна иметь следующие разделы:

1. Проект – раздел предназначен для описания задачи, которую Заказчик хочет запрограммировать.
2. Бюджет(необязательно) – раздел предназначен для указания определенной(диапазонной) суммы денег, которые Заказчик готов отдать за работу Исполнителя.
3. Скоки исполнения – в разделе Заказчик указывает срок исполнения поставленной задачи.
4. Дополнительно – раздел где Заказчик может указать особенности задачи, свои требования к поставленной задаче, критерий отбора Исполнителя.
5. Условия выполнения (заполняется Исполнителем и Заказчиком) – раздел, в котором Заказчик и Исполнитель договариваются о «тонкостях» разработки данной задачи :

    1. Условия невыполнения Исполнителем задачи в сроки, или же не отказ от задачи.
    2. Преждевременное выполнение задачи Исполнителем.
    3. Отказ Заказчика от выполненной задачи.

4.2 Процесс получения электронной заявки Исполнителем

Любой зарегистрированный фрилансер может предложить выполнение этой задачи и если Заказчику подойдет предложение данного фрилансера, Заказчик и Исполнитель заключают последний раздел в Электронной заявки Заказчика.

4.3 Возможности Заказчика

- Изменения своей заявки(до заполнения 5 раздела).

- Свободный выбор Исполнителя.
- Отправить жалобу Администратору на того или иного Исполнителя.
- Возможность размещения заказов.
- Вовлечения нескольких разработчиков в один проект.
- Оставлять комментарии на странице профиля разработчика о качестве его работы.

4.4 Возможности Исполнителя

- Мониторинг удачно/не удачно завершенных проектов
- Свободный выбор новых проектов на основе рейтинга.
- Оставлять комментарии на странице профиля заказчика о качестве приготовления им ТЗ, уровне профессионализма.
- Выкладывать примеры своих проектов.
- Юридическую защиту со стороны Администратора, в случае неоплаты работы

4.5 Возможности Администратора

- Принимать жалобы как со стороны Заказчик, так и со стороны Исполнителя, рассматривать их и принимать меры по урегулированию конфликта
- Удалять учётную запись Исполнителя за нарушения правил этикета.

5. Практичность

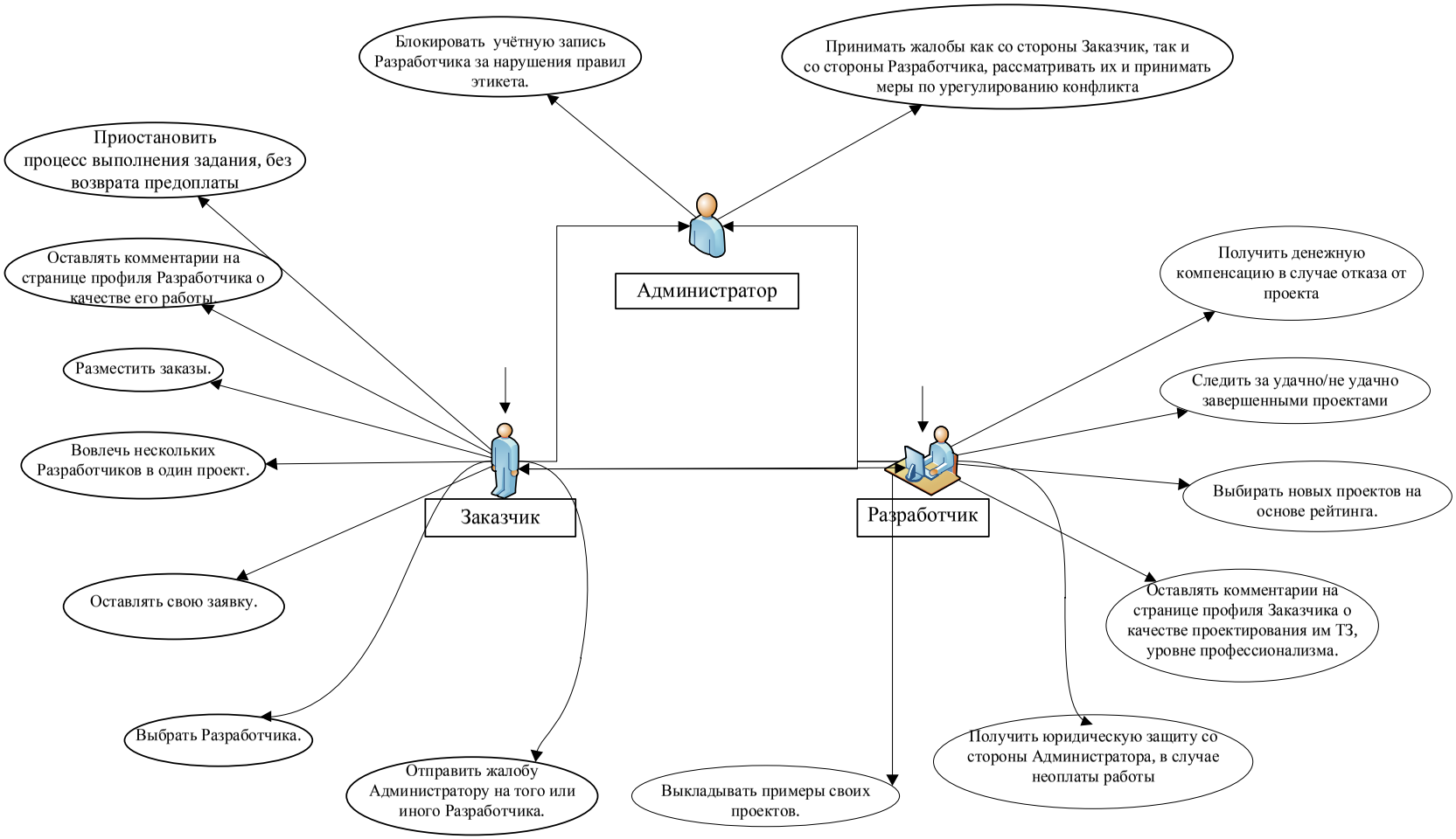
5.1 Интерфейс сайта

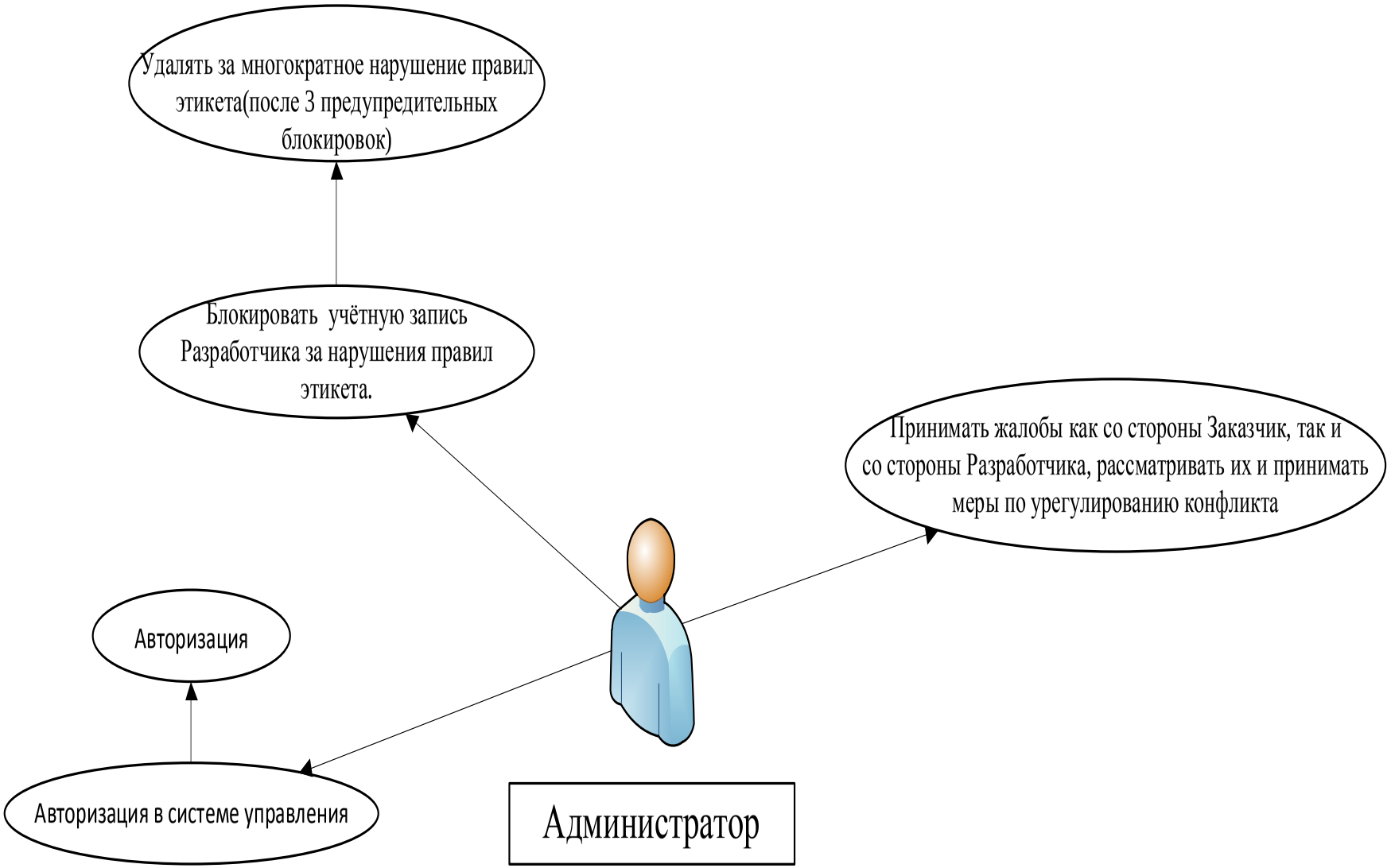
- Быть понятным и не допускать двузначного смысла
- Все зашифрованные параметры или элементы, сокращения, аббревиатуры должны иметь разъяснения в правом нижнем углу главной страницы сайта.

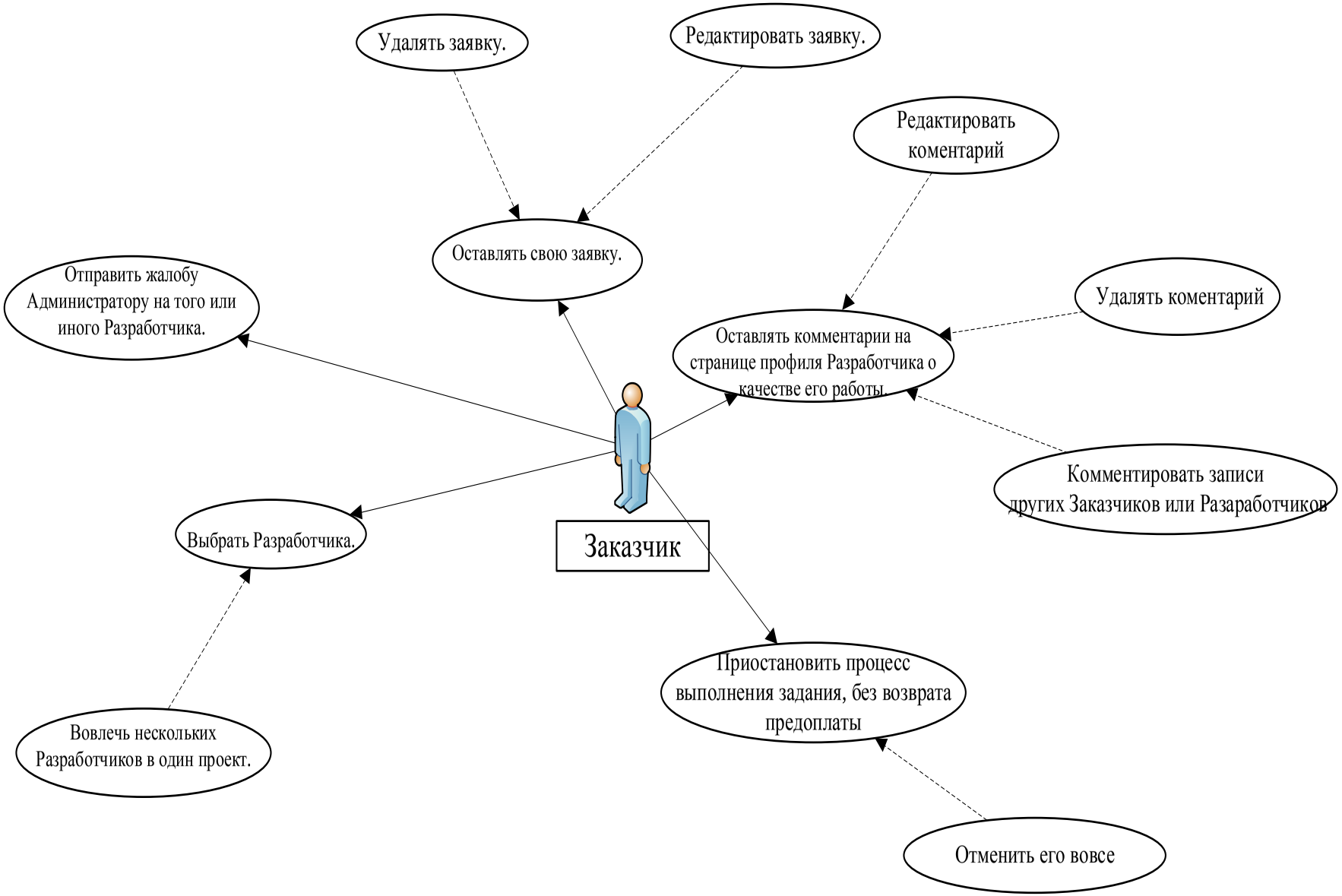
6. Надежность

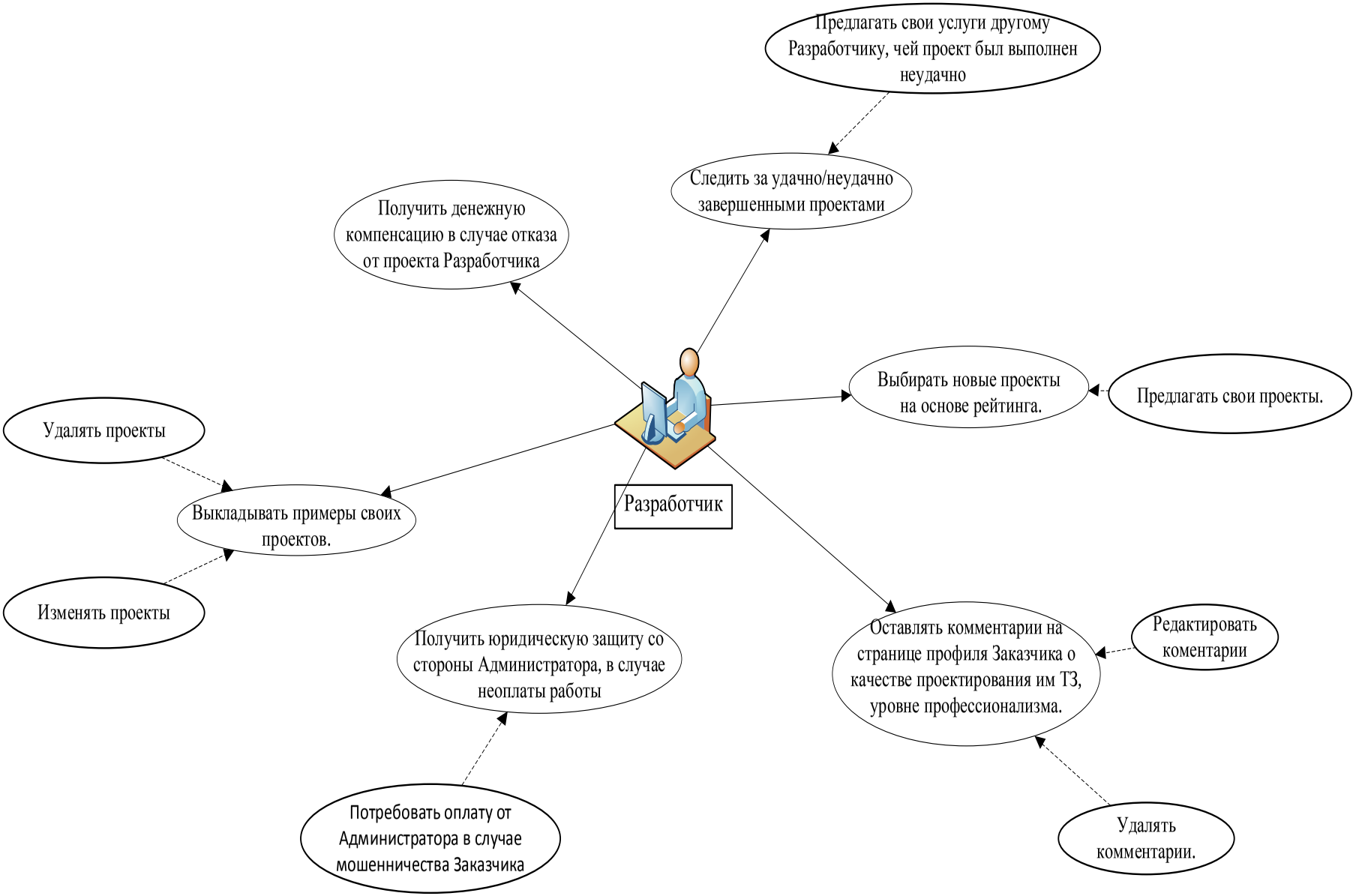
Все персональные данные Заказчиков и Исполнителей защищены и не могут быть использованы никем, только персоной, которой эти данные принадлежат.

Администрация сайта несет ответственность за нарушение пункта 5 заявки.


Блокировать учётную запись Разработчика за нарушения правил этикета.
Принимать жалобы как со стороны Заказчик, так и со стороны Разработчика, рассматривать их и принимать меры по урегулированию конфликта
Приостановить процесс выполнения задания, без возврата предоплаты
Оставлять комментарии на странице профиля Разработчика о качестве его работы.
Разместить заказы.
Вовлечь нескольких Разработчиков в один проект.
Оставлять свою заявку.
Выбрать Разработчика.
Отправить жалобу Администратору на того или иного Разработчика.
Администратор
Заказчик
Разработчик
Выкладывать примеры своих проектов.
Получить денежную компенсацию в случае отказа от проекта
Следить за удачно/не удачно завершенными проектами
Выбирать новых проектов на основе рейтинга.
Оставлять комментарии на странице профиля Заказчика о качестве проектирования им ТЗ, уровне профессионализма.
Получить юридическую защиту со стороны Администратора, в случае неоплаты работы

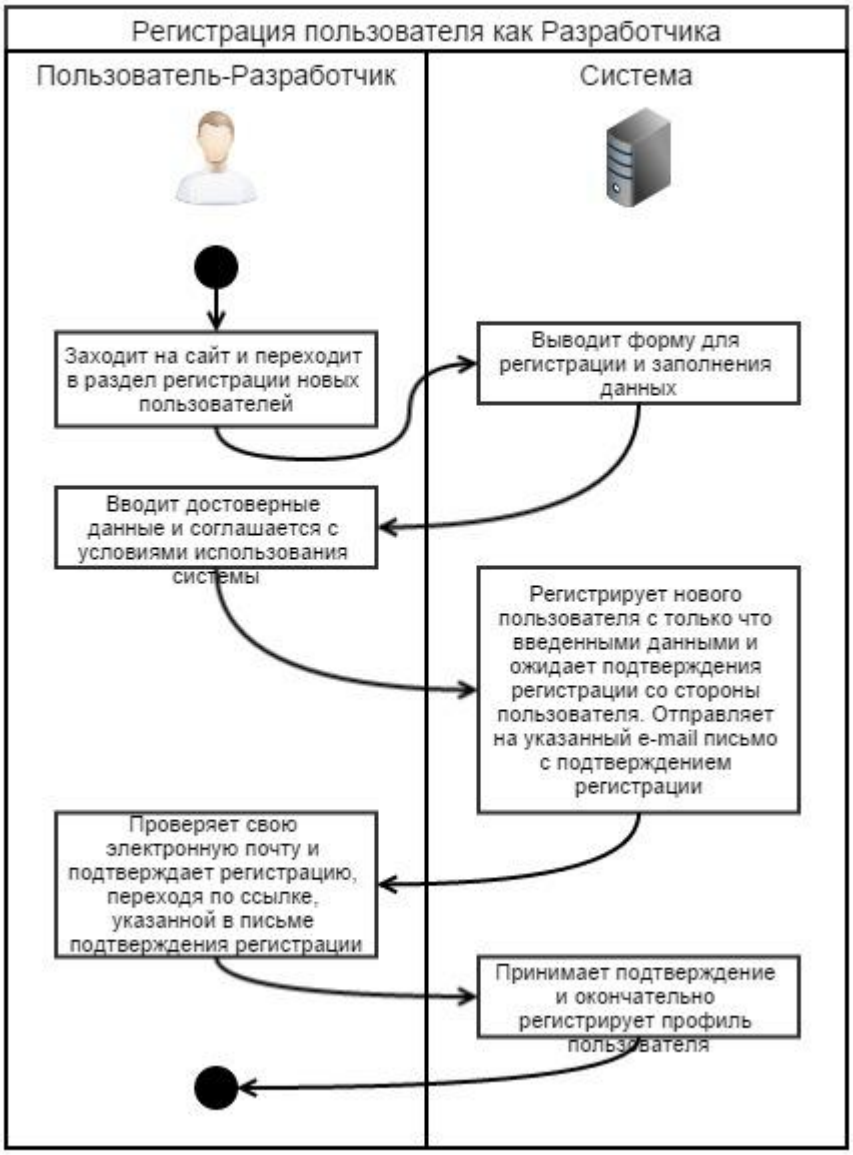

Удалять за многократное нарушение правил этикета(после 3 предупредительных блокировок)
Блокировать учётную запись Разработчика за нарушения правил этикета.
Принимать жалобы как со стороны Заказчик, так и со стороны Разработчика, рассматривать их и принимать меры по урегулированию конфликта
Авторизация
Авторизация в системе управления
Администратор


Удалять заявку.
Редактировать заявку.
Редактировать коментарий
Оставлять свою заявку.
Отправить жалобу Администратору на того или иного Разработчика.
Удалять коментарий
Оставлять комментарии на странице профиля Разработчика о качестве его работы.
Комментировать записи других Заказчиков или Разаработчиков
Выбрать Разработчика.
Заказчик
Приостановить процесс выполнения задания, без возврата предоплаты
Вовлечь нескольких Разработчиков в один проект.
Отменить его вовсе

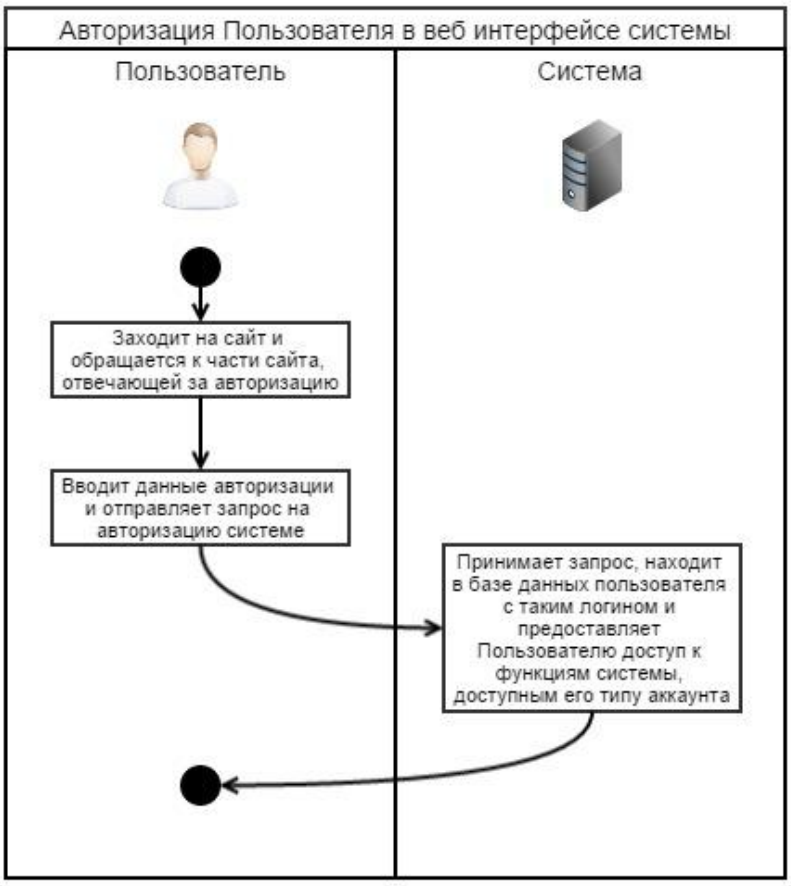

Предлагать свои услуги другому Разработчику, чей проект был выполнен неудачно
Следить за удачно/неудачно завершенными проектами
Получить денежную компенсацию в случае отказа от проекта Разработчика
Выбирать новые проекты на основе рейтинга.
Предлагать свои проекты.
Удалять проекты
Выкладывать примеры своих проектов.
Разработчик
Изменять проекты
Получить юридическую защиту со стороны Администратора, в случае неоплаты работы
Оставлять комментарии на странице профиля Заказчика о качестве проектирования им ТЗ, уровне профессионализма.
Редактировать коментарии
Потребовать оплату от Администратора в случае мошенничества Заказчика
Удалять комментарии.

Сценарии :

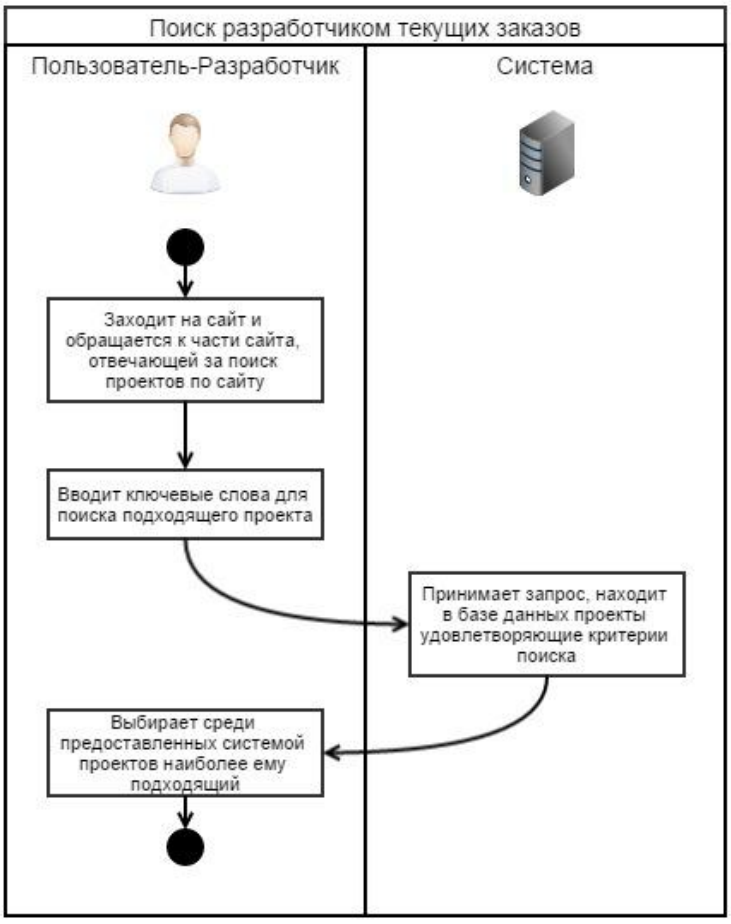
- **ID:** UC001
- **Название:** Регистрация пользователя как Разработчика.
- **Участники:** Пользователь-Разработчик, Система.
- **Предусловия:** Пользователь хочет стать новым Разработчиком, при этом не зарегистрирован в системе.
- **Результат:** Зарегистрирован новый пользователь, как Разработчик.
- **Основной сценарий:**
    1. Пользователь заходит на сайт и переходит в раздел регистрации новых пользователей.
    2. Система выводит форму для регистрации и заполнения данных.
    3. Пользователь вводит достоверные данные и соглашается с условиями использования системы.
    4. Система регистрирует нового пользователя с только что введенными данными и ожидает подтверждения регистрации со стороны пользователя. Отправляет на указанный e-mail письмо с подтверждением регистрации.
    5. Пользователь проверяет свою электронную почту и подтверждает регистрацию, переходя по ссылке, указанной в письме подтверждения регистрации.
    6. Система принимает подтверждение и окончательно регистрирует профиль пользователя.
- **Исключительные ситуации:**
    1. Пользователь не заполнил все необходимые поля.
    2. Пользователь указал неверный e-mail и не смог получить письмо подтверждения регистрацию.
    3. Пользователь не подтвердил регистрацию в течении 24 часов с отправки формы регистрации.


Регистрация пользователя как Разработчика
Пользователь-Разработчик
Система
Заходит на сайт и переходит в раздел регистрации новых пользователей
Выводит форму для регистрации и заполнения данных
Вводит достоверные данные и соглашается с условиями использования системы
Регистрирует нового пользователя с только что введенными данными и ожидает подтверждения регистрации со стороны пользователя. Отправляет на указанный e-mail письмо с подтверждением регистрации
Проверяет свою электронную почту и подтверждает регистрацию, переходя по ссылке, указанной в письме подтверждения регистрации
Принимает подтверждение и окончательно регистрирует профиль пользователя

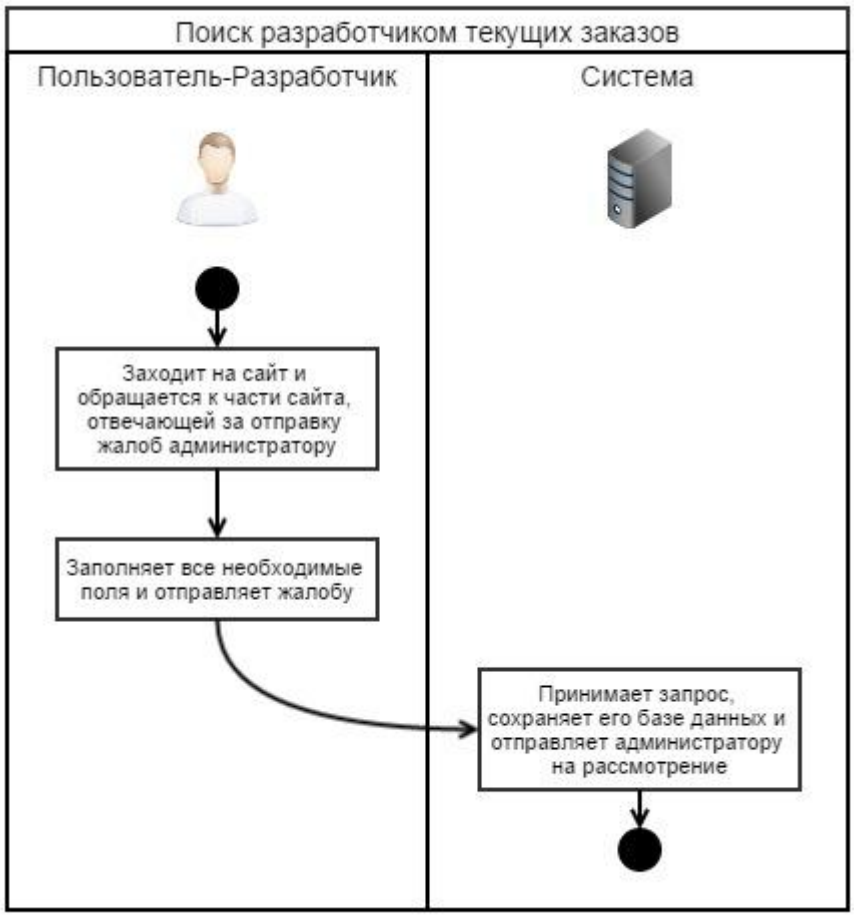
- **ID:** UC002
- **Название:** Авторизация Пользователя в веб интерфейсе системы.
- **Участники:** Пользователь, Система.
- **Предусловия:** Пользователь не вошел в систему и не имеет доступа к функциям системы.
- **Результат:** Пользователь получает доступ к функциям системы и своему профилю.
- **Основной сценарий:**
    1. Пользователь заходит на сайт и обращается к части сайта, отвечающей за авторизацию.
    2. Пользователь вводит данные авторизации и отправляет запрос на авторизацию системе.
    3. Система принимает запрос, находит в базе данных пользователя с таким логином и предоставляет Пользователю доступ к функциям системы, доступным его типу аккаунта.
- **Исключительные ситуации:**
    1. Пользователь ввел неверные данные.
    2. Система временно недоступна/произошел сбой в системе.



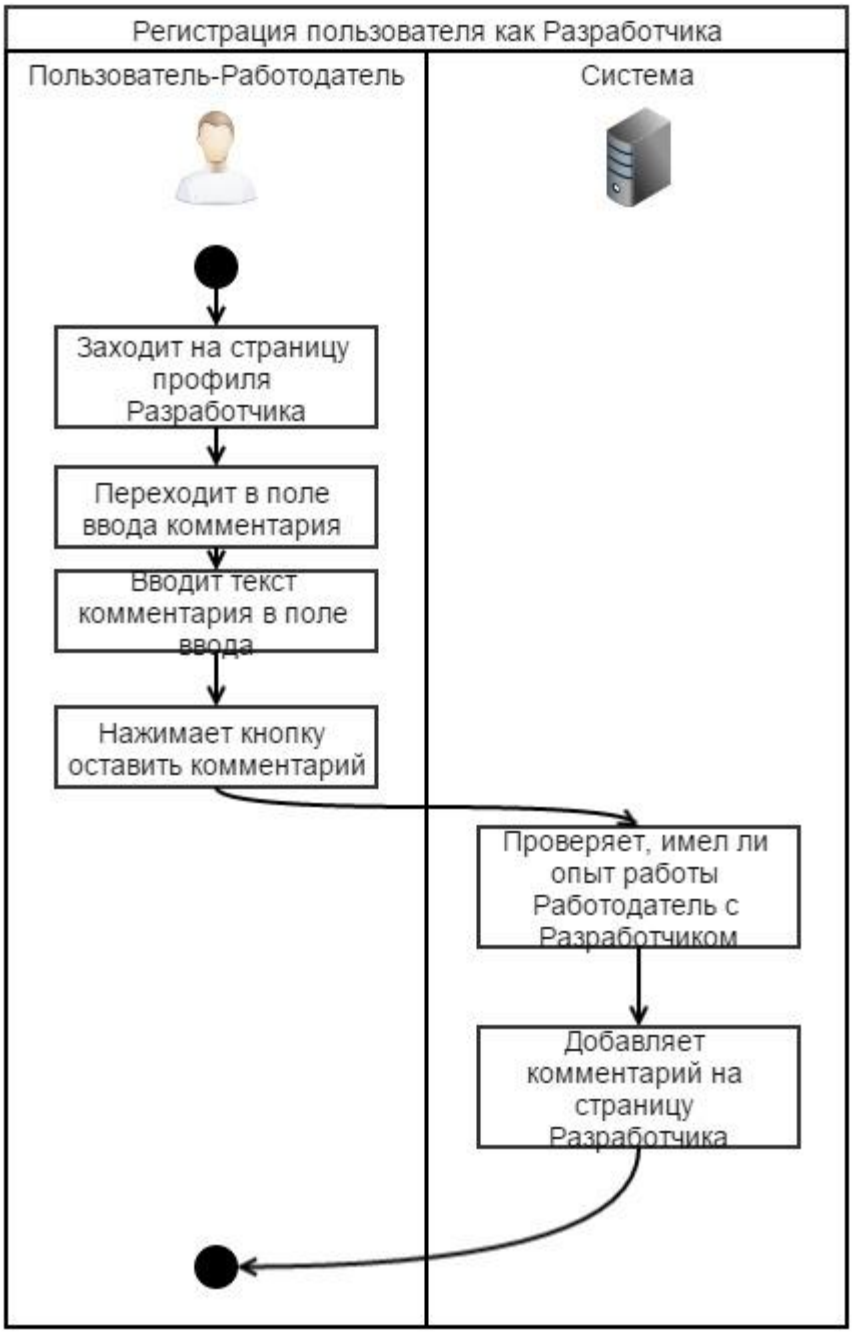
- **ID:** UC003
- **Название:** Поиск разработчиком заказов.
- **Участники:** Пользователь, Система.
- **Предусловия:** Пользователь вошел в систему и имеет доступа к функциям системы.
- **Результат:** Пользователь находит подходящий заказ.
- **Основной сценарий:**
    1. Пользователь заходит на сайт и обращается к части сайта, отвечающей за поиск проектов по сайту.
    2. Пользователь вводит ключевые слова для поиска подходящего проекта.
    3. Система принимает запрос, находит в базе данных проекты удовлетворяющие критерии поиска.
    4. Пользователь выбирает среди предоставленных системой проектов наиболее ему подходящий.
- **Исключительные ситуации:**
    1. Проект удовлетворяющий критерии поиска не существует.
    2. Пользователь допустил ошибку в запросе.



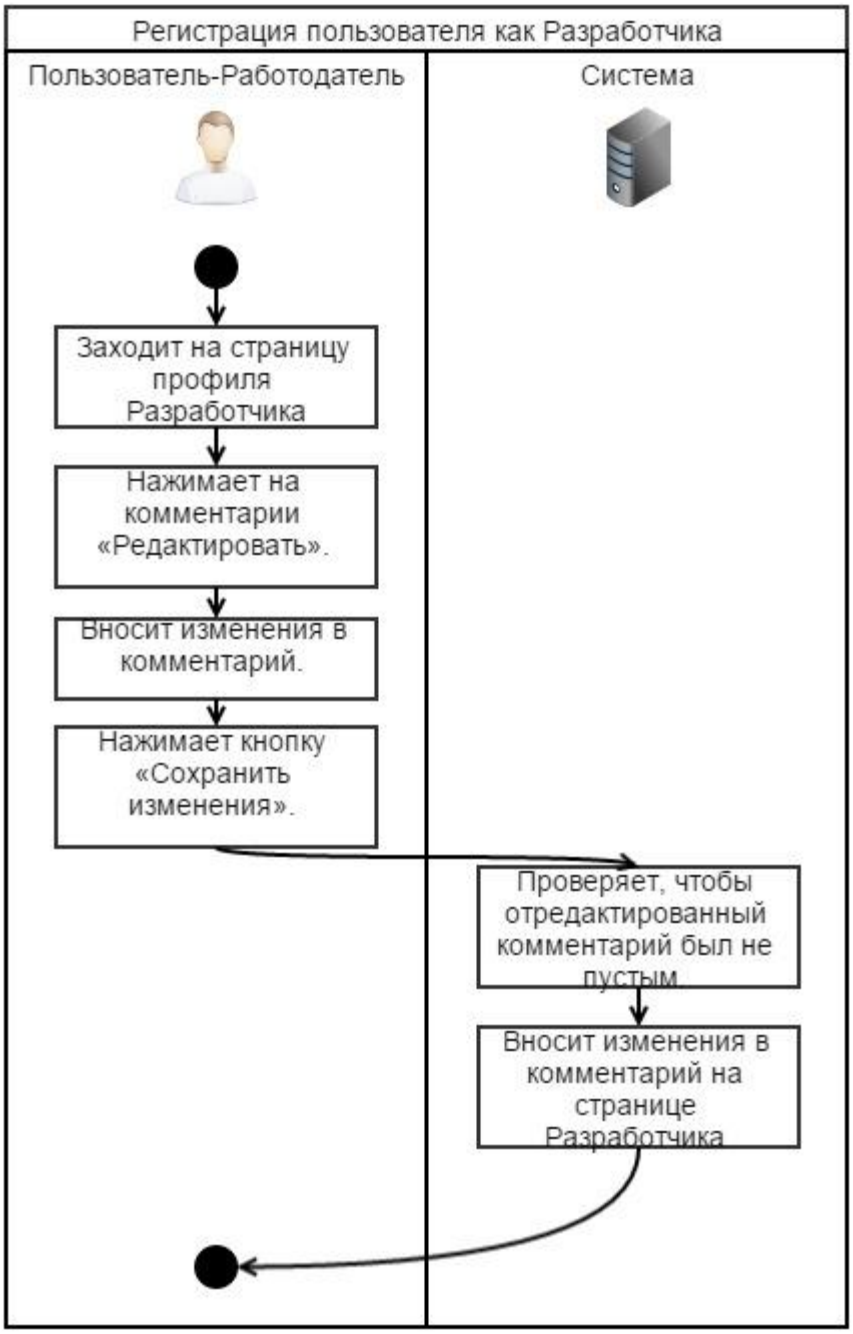
- **ID:** UC004
- **Название:** Подача жалобы.
- **Участники:** Пользователь, Система.
- **Предусловия:** Пользователь вошел в систему и имеет доступ к функциям системы.
- **Результат:** Пользователь отправляет жалобу.
- **Основной сценарий:**
    1. Пользователь заходит на сайт и обращается к части сайта, отвечающей за отправку жалоб администратору.
    2. Пользователь заполняет все необходимые поля и отправляет жалобу.
    3. Система.
- **Исключительные ситуации:**
    1. Пользователь не заполнил все необходимые поля.



- **ID:** UC005
- **Название:** Добавление комментария на странице разработчика о качестве его работы.
- **Участники:** Работодатель, Система.
- **Предусловия:** Работодатель имел опыт работы с Разработчиком и хочет оставить комментарий о качестве выполнения работы
- **Результат:** На странице профиля разработчика появляется новый комментарий
- **Основной сценарий:**
    1. Работодатель заходит на страницу профиля Разработчика.
    2. Работодатель переходит в поле ввода комментария
    3. Работодатель вводит текст комментария в поле ввода.
    4. Работодатель нажимает кнопку оставить комментарий.
    5. Система проверяет, имел ли опыт работы Работодатель с Разработчиком.
    6. Система добавляет комментарий на страницу Разработчика
- **Исключительные ситуации:**
    1. Работодатель хочет оставить комментарий на странице неизвестного ему Разработчика.

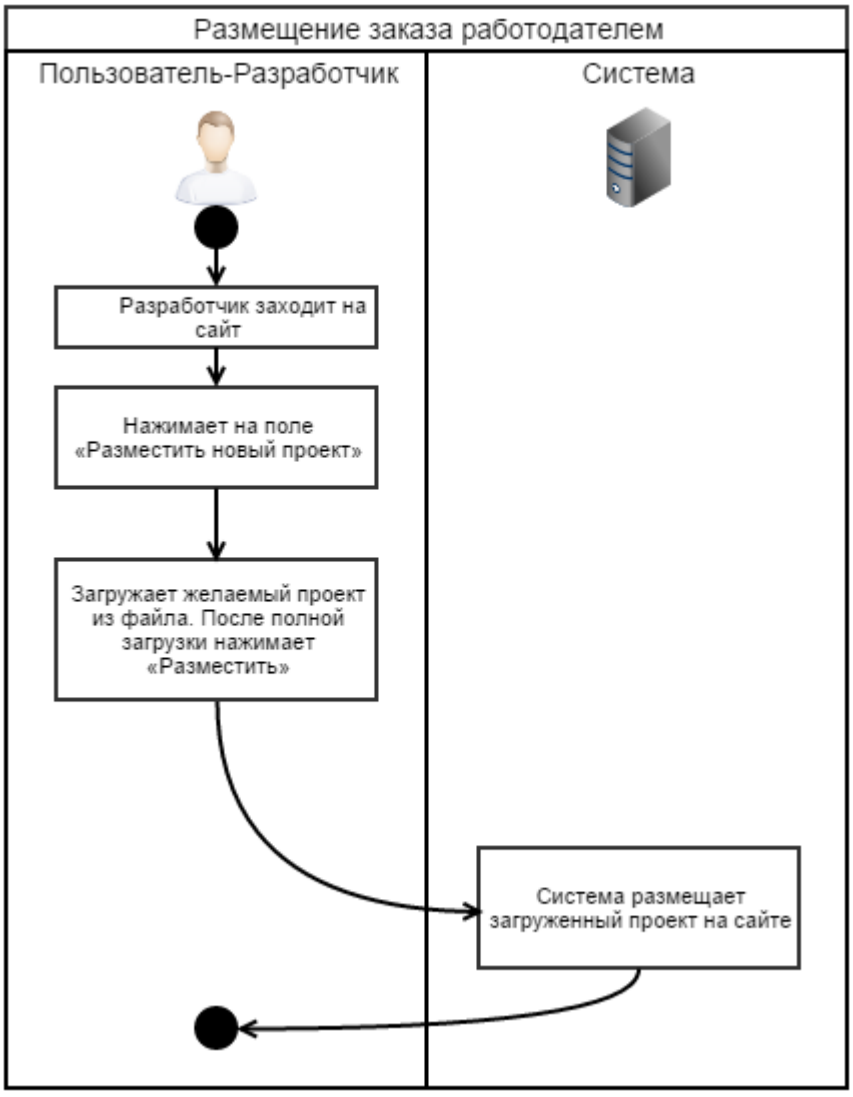

Регистрация пользователя как Разработчика
Пользователь-Работодатель
Система
Заходит на страницу профиля Разработчика
Переходит в поле ввода комментария
Вводит текст комментария в поле ввода
Нажимает кнопку оставить комментарий
Проверяет, имел ли опыт работы Работодатель с Разработчиком
Добавляет комментарий на страницу Разработчика

- **ID:** UC006
- **Название:** Редактирование комментария на странице разработчика.
- **Участники:** Работодатель, Система.
- **Предусловия:** Работодатель оставил комментарий о Разработчике и хочет его отредактировать
- **Результат:** Комментарий, оставленный Работодателем на странице профиля Разработчика, отредактирован
- **Основной сценарий:**
    7. Работодатель заходит на страницу профиля Разработчика.
    8. Работодатель нажимает на комментарии «Редактировать».
    9. Работодатель вносит изменения в комментарий.
    10. Работодатель нажимает кнопку «Сохранить изменения».
    11. Система проверяет, чтобы отредактированный комментарий был не пустым.
    12. Система вносит изменения в комментарий на странице Разработчика
- **Исключительные ситуации:**
    2. Работодатель оставляет поле ввода текста пустым.

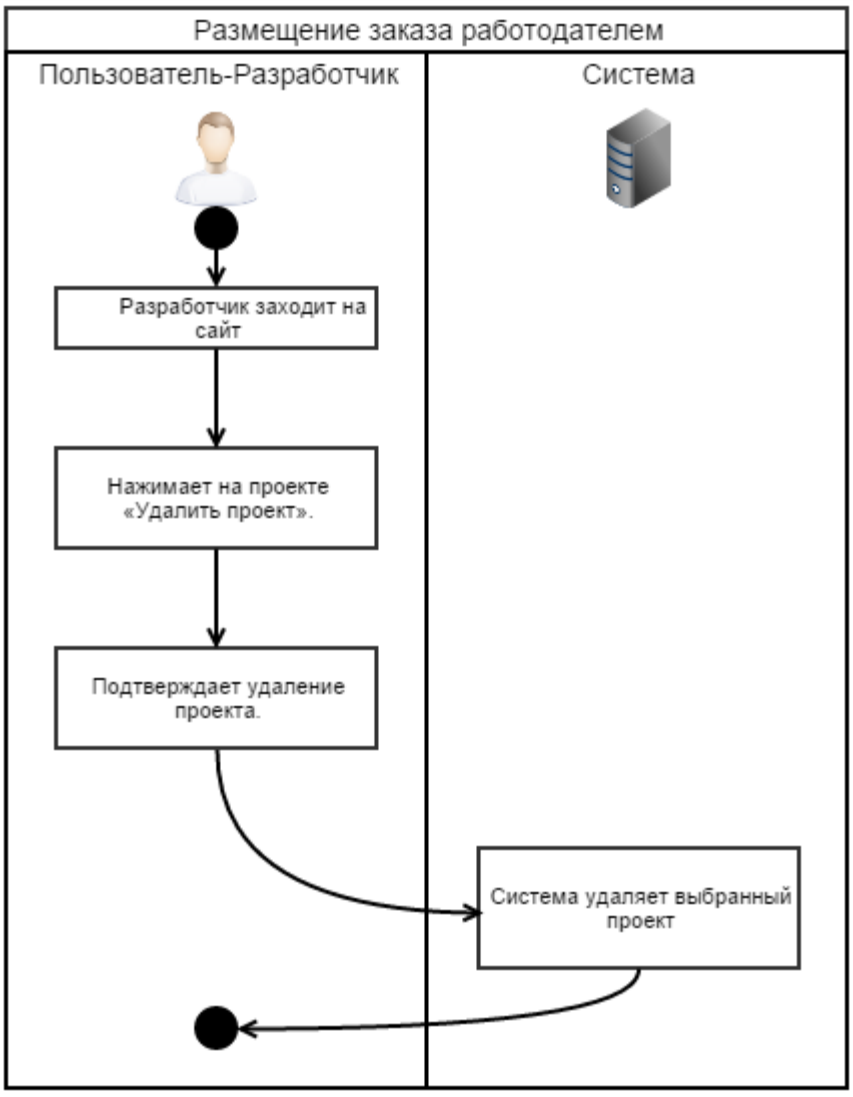

Регистрация пользователя как Разработчика
Пользователь-Работодатель
Система
Заходит на страницу профиля Разработчика
Нажимает на комментарии «Редактировать».
Вносит изменения в комментарий.
Нажимает кнопку «Сохранить изменения».
Проверяет, чтобы отредактированный комментарий был не пустым.
Вносит изменения в комментарий на странице Разработчика

- **ID:** UC007
- **Название:** Удаление комментария на странице разработчика.
- **Участники:** Работодатель, Система.
- **Предусловия:** Работодатель оставил комментарий о Разработчике и хочет его удалить
- **Результат:** Комментарий, оставленный Работодателем на странице профиля Разработчика, удален
- **Основной сценарий:**
    13. Работодатель заходит на страницу профиля Разработчика.
    14. Работодатель нажимает на комментарии «Удалить».
    15. Работодатель подтверждает удаление.
    16. Система удаляет комментарий со страницы Разработчика
- **Исключительные ситуации:**
    3. Работодатель отменяет удаление комментария.

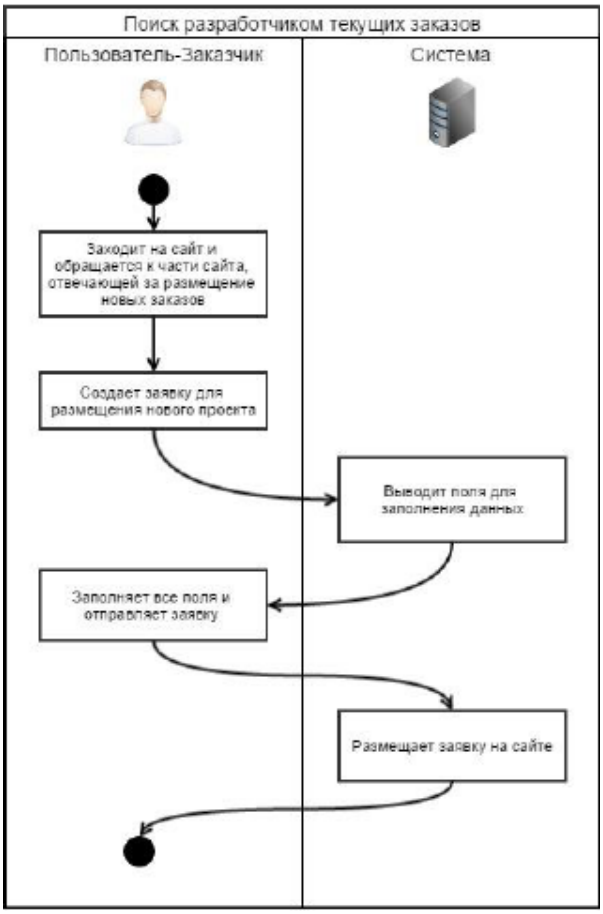
- **ID:** UC008
- **Название:** Размещение проекта разработчиком.
- **Участники:** Разработчик, Система.
- **Предусловия:** Разработчик хочет выложить на всеобщее обозрение свой готовый проект.
- **Результат:** Разработчик размещает на сайте желаемый проект.
- **Основной сценарий:**
    1. Разработчик заходит на сайт.
    2. Нажимает на поле «Разместить новый проект».
    3. Загружает желаемый проект из файла. После полной загрузки нажимает «Разместить»
    4. Система размещает загруженный проект на сайте.
- **Исключительные ситуации:**
    1. У файла слишком большой размер, что не позволит Разработчику разместить его.



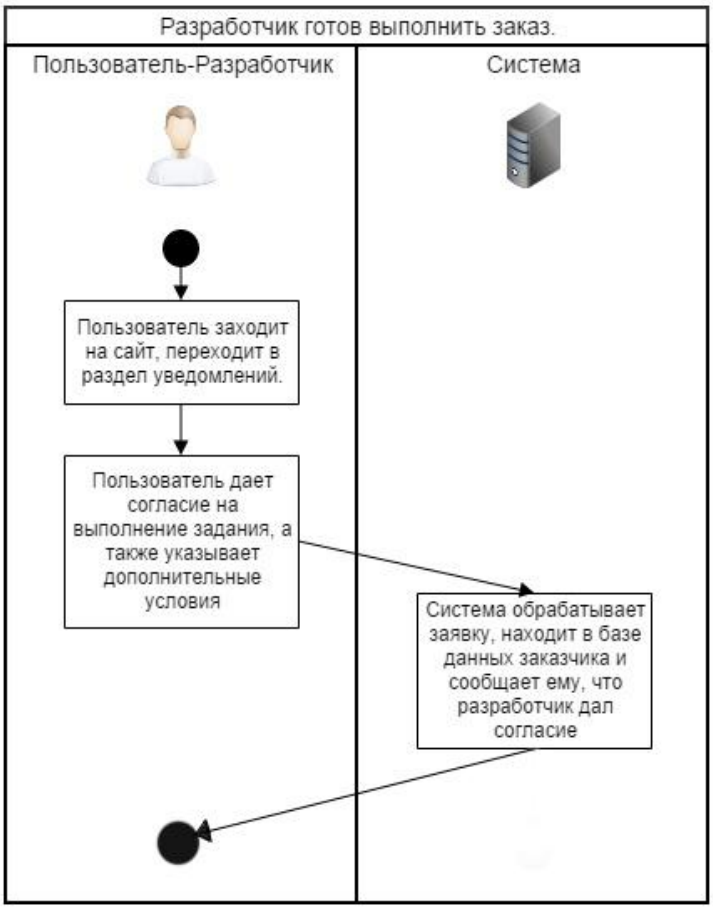
- **ID:** UC009
- **Название:** Удаление проекта разработчиком.
- **Участники:** Разработчик, Система.
- **Предусловия:** Разработчик хочет удалить свой проект, который размещен на сайте.
- **Результат:** удаленный проект.
- **Основной сценарий:**
    1. Разработчик заходит на сайт.
    2. Нажимает на проекте «Удалить проект».
    3. Подтверждает удаление проекта.
    4. Система удаляет выбранный проект.
- **Исключительные ситуации:**
    1. У разработчика нет выложенных проектов.



- **ID:** UC0010
- **Название:** Размещение заказа работодателем.
- **Участники:** Работодатель, Система.
- **Предусловия:** Пользователь вошел в систему и имеет доступ к функциям
- системы.
- **Результат:** Работодатель размещает заказ.
- **Основной сценарий:**
    1. Пользователь заходит на сайт и обращается к части сайта, отвечающей за
       размещение новых заказов.
    2. Пользователь создает заявку для размещения нового проекта.
    3. Система выводит поля для заполнения данных.
    4. Пользователь заполняет все поля и отправляет заявку.
    5. Система размещает заявку на сайте.
- **Исключительные ситуации:**
    1. Пользователь не заполнил все поля.



- **ID:** UC011
- **Название:** Разработчик готов выполнить заказ.
- **Участники:** Пользователь-разработчик, Система.
- **Предусловия:** Пользователь – разработчик вошел в систему и имеет доступа к функциям системы.
- **Результат:** Пользователь – разработчик принимает на выполнение заказ.
- **Основной сценарий:**
    1. Пользователь заходит на сайт, переходит в раздел уведомлений.
    2. Пользователь дает согласие на выполнение задания, а также указывает дополнительные условия.
    3. Система обрабатывает заявку, находит в базе данных заказчика и сообщает ему, что разработчик дал согласие.
-
- **Исключительные ситуации:**
    1. Аккаунт пользователя – заказчика уже не существует (удален).



- **ID:** UC012
- **Название:** Добавление отзыва**.**
- **Участники:** Пользователь , Система.
- **Предусловия:** Пользователь вошел в систему и имеет доступ ее функциям
- **Результат:** Оставлен отзыв об работодателе или разработчике.
- **Основной сценарий:**
    1. Пользователь заходит на сайт, переходит на аккаунт другого пользователя.
    2. Пользователь переходить в раздел «Отзывы ».
    3. Создает отзыв и соглашается с его публикацией.
    4. Система обрабатывает отзыв и делает запись в разделе «Отзывы» пользователя , которому оставили отзыв .

- **Исключительные ситуации:**
    1. Отзыв содержит нецензурную лексику, оскорбление.

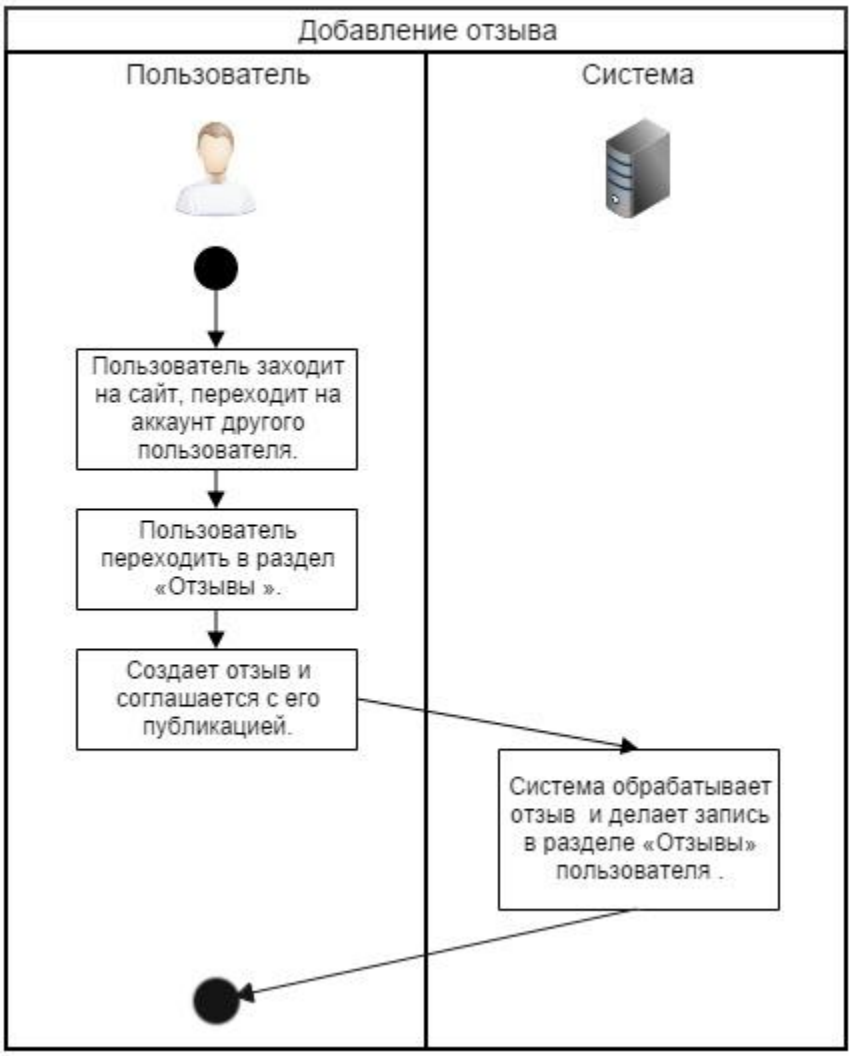
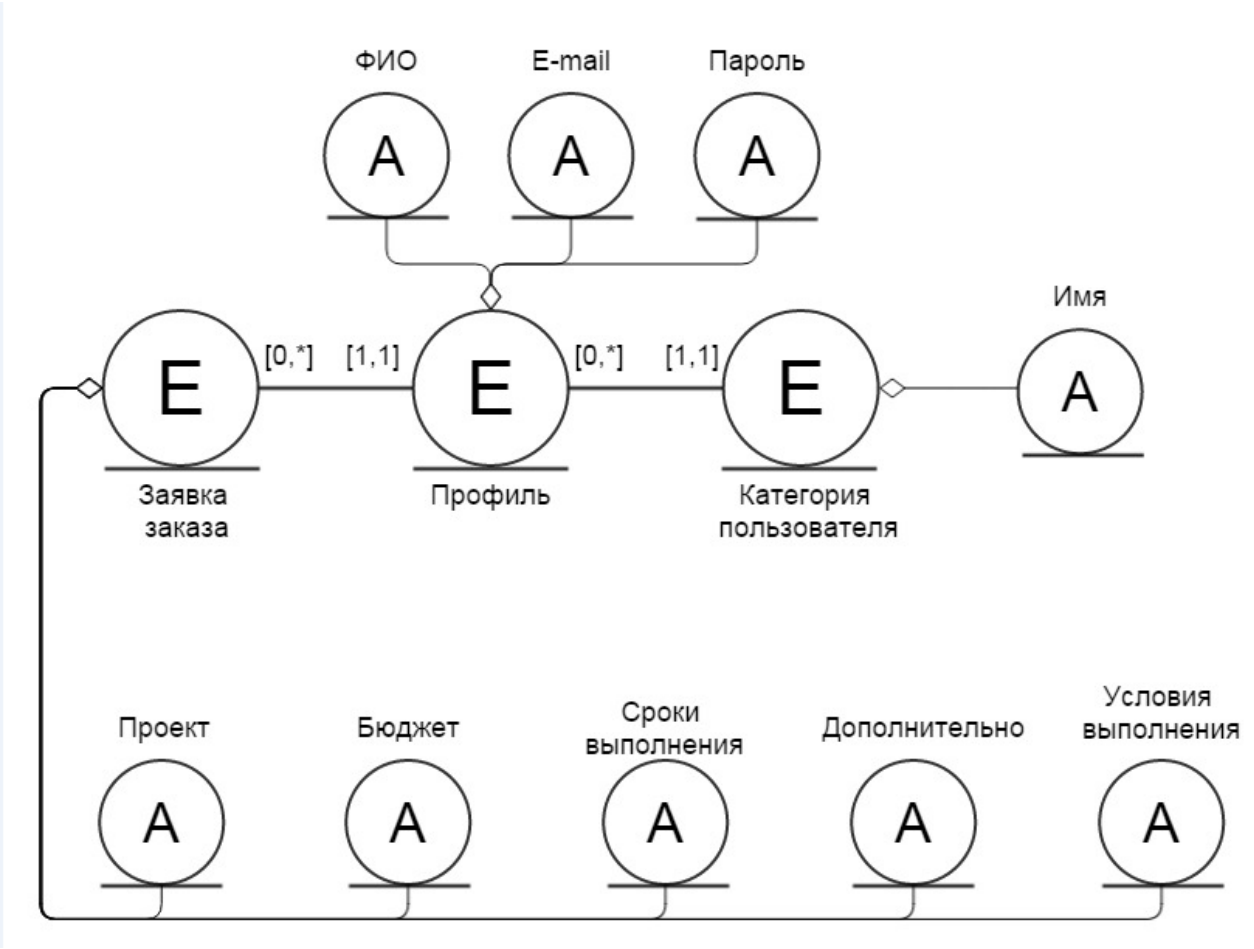


Диаграмма сущностей:



Профиль

Название	Тип	Ограничение
ФИО	String	
e-mail	String	
Password	String	
Заявка	Number	FK
Категория Пользователя	Number	FK
ID	Number	FK

[1,1]

[0,*]

Категория пользователя

Название	Тип	Ограничение
Название категории	String	
ID	Number	FK

[1,1]

Заявка заказа

Название	Тип	Ограничение
Проект	String	
Бюджет	Number	
Сроки выполнения	Date	
Дополнительно	String	
Условие выполнения	String	
ID	Number	FK

[0,*]

Представление о пользователе

Профиль – внутреннее представление пользователя, которое задаётся атрибутами:

- ФИО;
- E-mail;
- Пароль.

Категория пользователя - совокупность людей, которые обладают общими правами, ролью, возможностями, доступом. Задаётся атрибутом:

- Тип пользователя.

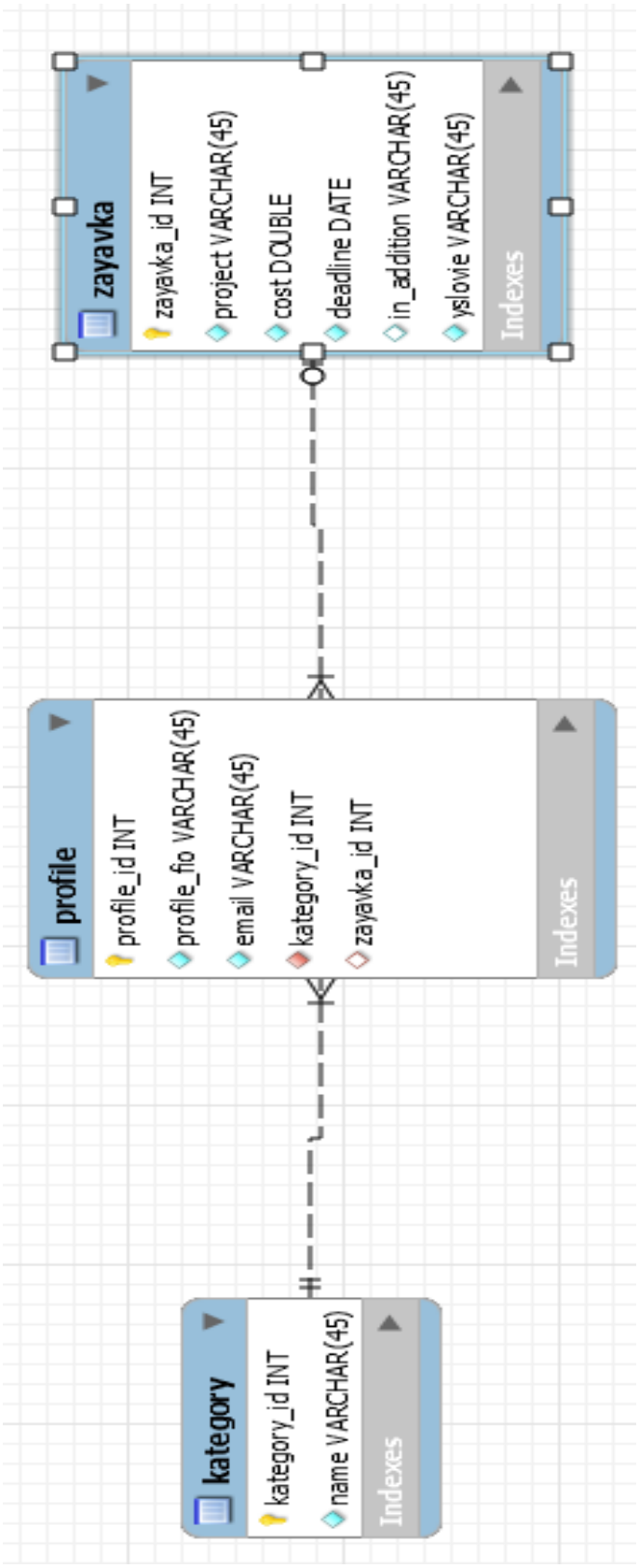
Заявка заказа- подается Заказчиком, задаётся атрибутами:

- Проект;
- Бюджет;
- Сроки исполнения;
- Дополнительно;
- Условия выполнения.

Атрибуты

1. ФИО – атрибут профиля;
2. Пароль – атрибут профиля;
3. E-mail – атрибут профиля;
4. Тип пользователя – атрибут категории пользователя;
5. Проект – атрибут заявки заказа;
6. Бюджет – атрибут заявки заказа;
7. Сроки исполнения – атрибут заявки заказа;
8. Дополнительно – атрибут заявки заказа;
9. Условия выполнения – атрибут заявки заказа.

Mysql workbench диаграмма:



Листинг программы:

```
package database;

/**
 * Created by group №5 on 11/20/2014.
 */

import com.mysql.fabric.jdbc.FabricMySQLDriver;

import java.sql.*;

public class DatabaseConection {
    private static final String URL= "jdbc:mysql://localhost:3306/freelance";
    private static final String LOGIN= "root";
    private static final String PASS= "root";

    private Connection connection;

    public static void main(String[] args) {


        DatabaseConection dc = new DatabaseConection();
        dc.establishConnection();



        dc.printTable("profile");
        //String[] cellNames = {"profile_id", "profile_fio", "email", "kategory_id", "zayavka_id"};

        //dc.update("profile", 2, new String[]{"profile_fio"}, new Object[]{"'Dovgal'"});

        System.out.println();
        dc.printTable("zayavka");

    }


    public void establishConnection() {
        try {
            DriverManager.registerDriver(new FabricMySQLDriver());
        } catch (SQLException e) {
            System.err.printf("Driver wasn't loaded");
        }

        try {
```

```
            connection = DriverManager.getConnection(URL, LOGIN, PASS);
        }catch (SQLException e){
            e.printStackTrace();
        }
    }

    public void printTable(String tableName){

        if(!tableName.equals("profile") && !tableName.equals("zayavka") &&
!tableName.equals("kategory")){
            System.out.println("Wrong table name");
            return;
        }


        try {
            java.sql.Statement statement= connection.createStatement();




            ResultSet columnsCounter = statement.executeQuery("show columns from freelance." + tableName
+";");

            int counter = 0;
            while (columnsCounter.next()){

                //System.out.print(columnsCounter.getString(1) + " ");
                counter++;

            }
            System.out.println();

            ResultSet resultSet= statement.executeQuery("select * from freelance." + tableName + ";");

            while (resultSet.next()){
                for(int i = 1; i <= counter;i++){
                    System.out.print(resultSet.getString(i) + " ");
                }
                System.out.println();

            }


        }catch (SQLException se){
            System.out.println("Wrong query");
```

```
        }
    }



    public void insert(String tableName, String[] cellNames, Object[] values){


        if(!tableName.equals("profile") && !tableName.equals("zayavka") &&
!tableName.equals("kategory")){
            System.out.println("Wrong table name");
            return;
        }

        try {
            java.sql.Statement statement= connection.createStatement();


            StringBuilder querry = new StringBuilder("insert into freelance." + tableName+ " (");

            for(int i = 0; i < cellNames.length-1; i++){
                querry.append(cellNames[i] + ",");
            }
            querry.append(cellNames[cellNames.length-1] + ") value (");
            for(int i = 0; i < values.length-1; i++){
                querry.append(values[i] + ",");
            }
            querry.append(values[values.length-1] + ");");



            statement.execute(querry.toString());
        }catch (SQLException se){
            System.out.println("Wrong query");
        }


    }

    public void delete(String tableName, int cellId){

        if(!tableName.equals("profile") && !tableName.equals("zayavka") &&
!tableName.equals("kategory")){
            System.out.println("Wrong table name");
            return;
        }
```

```
        try {
            java.sql.Statement statement= connection.createStatement();

            statement.execute("delete from freelance."+tableName+" where "+tableName+"_id=" + cellId + ";");


        }catch (SQLException se){
            System.out.println("Wrong query");
        }
    }

    public void update(String tableName, int cellId, String[] cellNames, Object[] values){

        if(!tableName.equals("profile") && !tableName.equals("zayavka") &&
!tableName.equals("kategory")){
            System.out.println("Wrong table name");
            return;
        }

        try {
            java.sql.Statement statement= connection.createStatement();

            StringBuilder querry = new StringBuilder("update freelance." + tableName+ " set ");

            for(int i = 0; i < cellNames.length; i++){
                querry.append(cellNames[i] + "=" + values[i].toString() + " ");
            }

            querry.append("where " + tableName + "_id=" + cellId + ";");



            statement.execute(querry.toString());
        }catch (SQLException se){
            System.out.println("Wrong query");
        }
    }


}
```