**Анализ предметной области.**

Trello – веб-приложение для управления проектами, популярное среди небольших групп разработчиков или групп с ограниченным бюджетом. Это приложение бесплатно и предоставляет меньше функций чем другие платные аналоги. TrelloSuggest стремится расширить функционал Trello путем добавления анализа загруженности и предложения вариантов распределения загруженности. Наиболее популярными аналогами Trello являются: Clarizen, Paymo, Gitscrum, Blossom, Jira.

**Clarizen.** Крупное приложение, предоставляющее много статистики, включая нагруженность, однако нагруженность присутствует только в демонстративном виде и советов по перераспределению не дается. Кроме того, приложение является платным.

**Paymo.** Другая альтернатива Trello с большим фокусом на автоматическое отслеживание времени, предоставляющее возможность бесплатного пользования одному пользователю. TrelloSuggest фокусируется на нагруженности, а следовательно на множестве пользователей, что и отличает его от Paymo.

**Gitscrum.** Данное веб-приложение предоставляет схожий с Trello функционал, однако модуль расчета загруженности в этот функционал не входит. К тому же, у Gitscrum отсутствует мобильное приложение.

**Blossom.** Приложение не предлагает возможностей TrelloSuggest, аналитический модуль фокусируется на рассчете производительности, не имеется мобильного приложения. Также Blossom не обладает бесплатной версией.

**Jira.** Предоставляет очень много разнообразных возможностей и услух, в том числе аналитику и отчеты. Многое из этого направлено на производительность, но не связано с основным назначением TrelloSuggest.

Таким образом, функционал TrelloSuggest представлен в небольшом числе приложений и, если представлен, то только частично. Кроме того, бесплатность Trello позволяет предоставить более высокий уровень доступности.

Так как Trello распространяется как в мобильной версии, так и в десктопной, было принято решение разработать приложение для Android и веб-интерфейс.

Объектная иерархия Trello:

1. **Доска** – проект, над которым работает пользователь. Содержит список списков.
2. **Список** – контейнер для карточек, определяющий их статус.
3. **Карточка** – единичное задание, которое может быть назначено одному или нескольким пользователям.

Исходя из этой иерархии, интерфейс пользователя TrelloSuggest

организован следующим образом – после авторизации пользователь попадает на страницу досок. Оттуда он может сразу перейти к загруженности или посмотреть определенные задачи. Это отражено на диаграммах активности, состояния и вариантов использования:

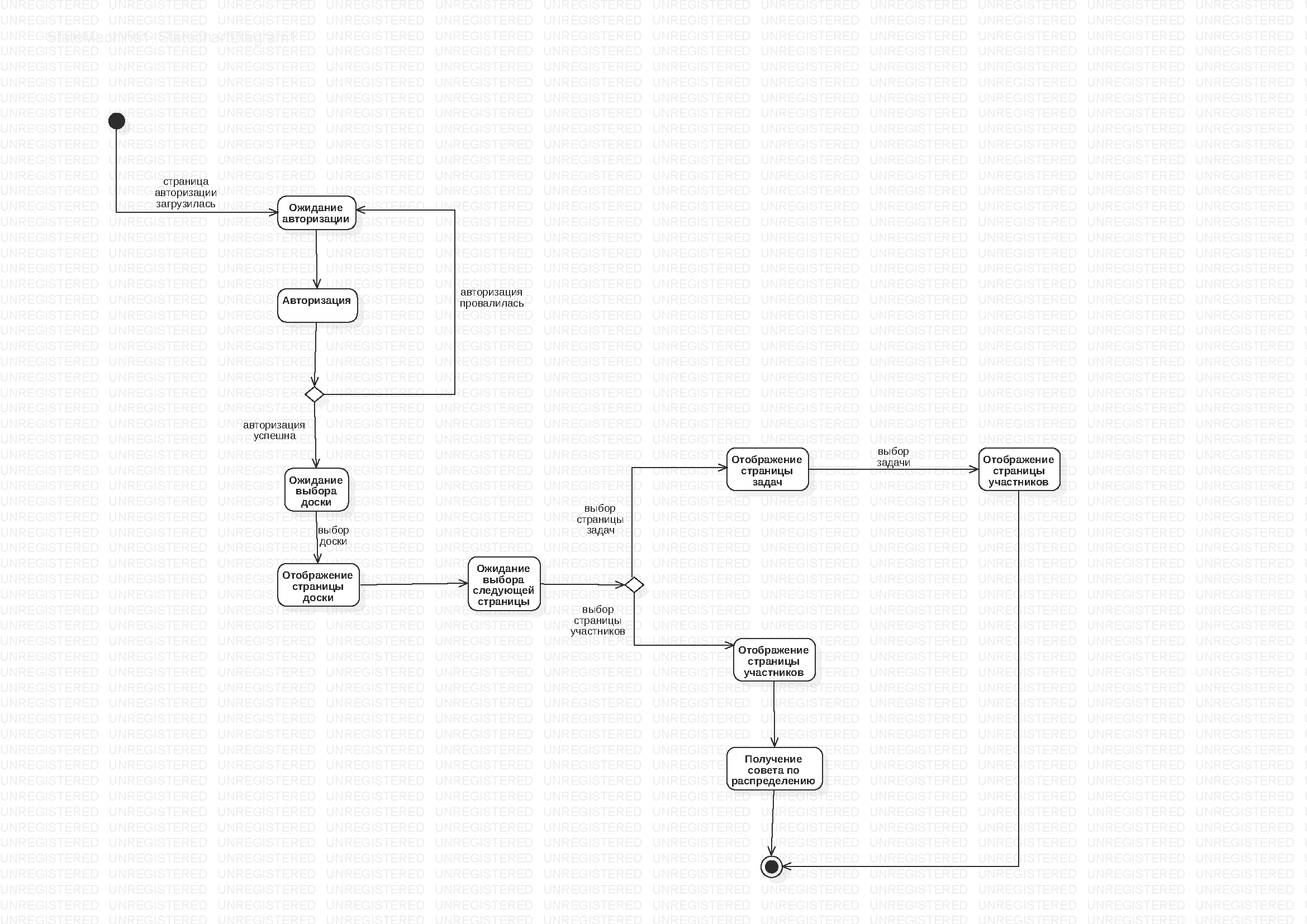


Диаграмма состояния

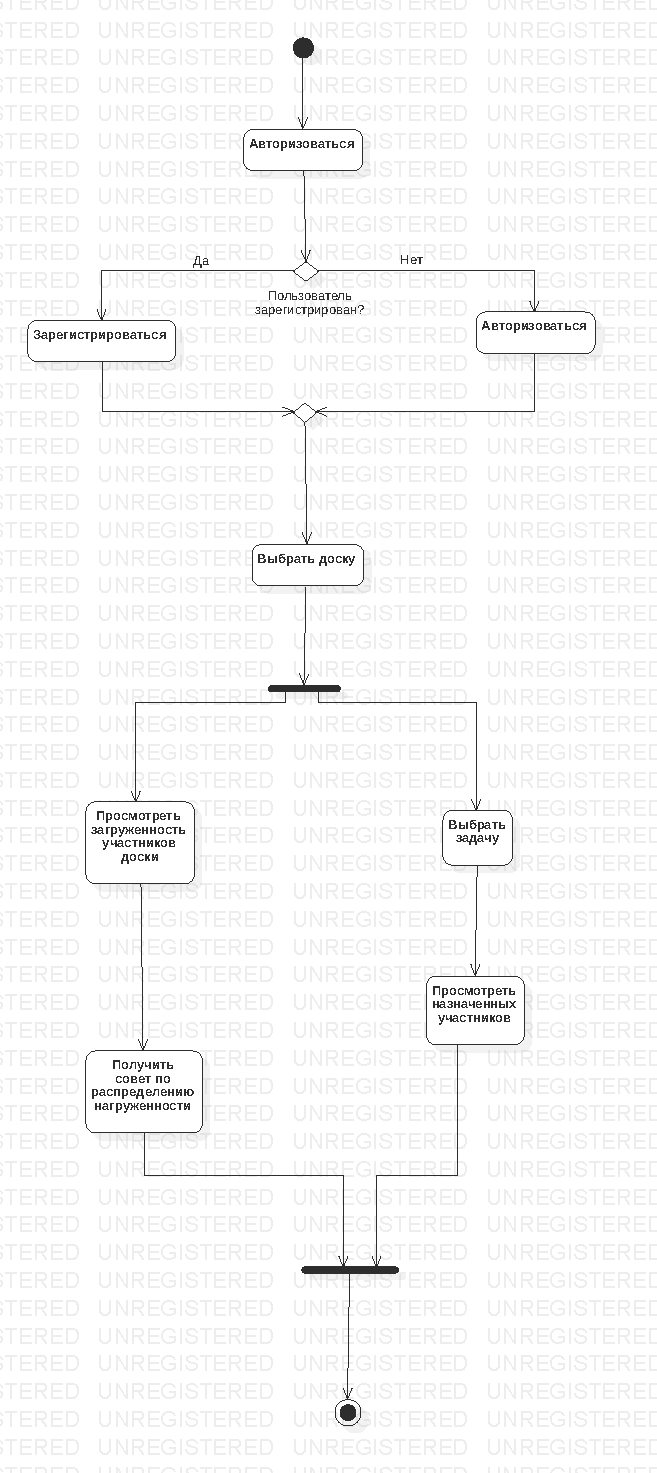
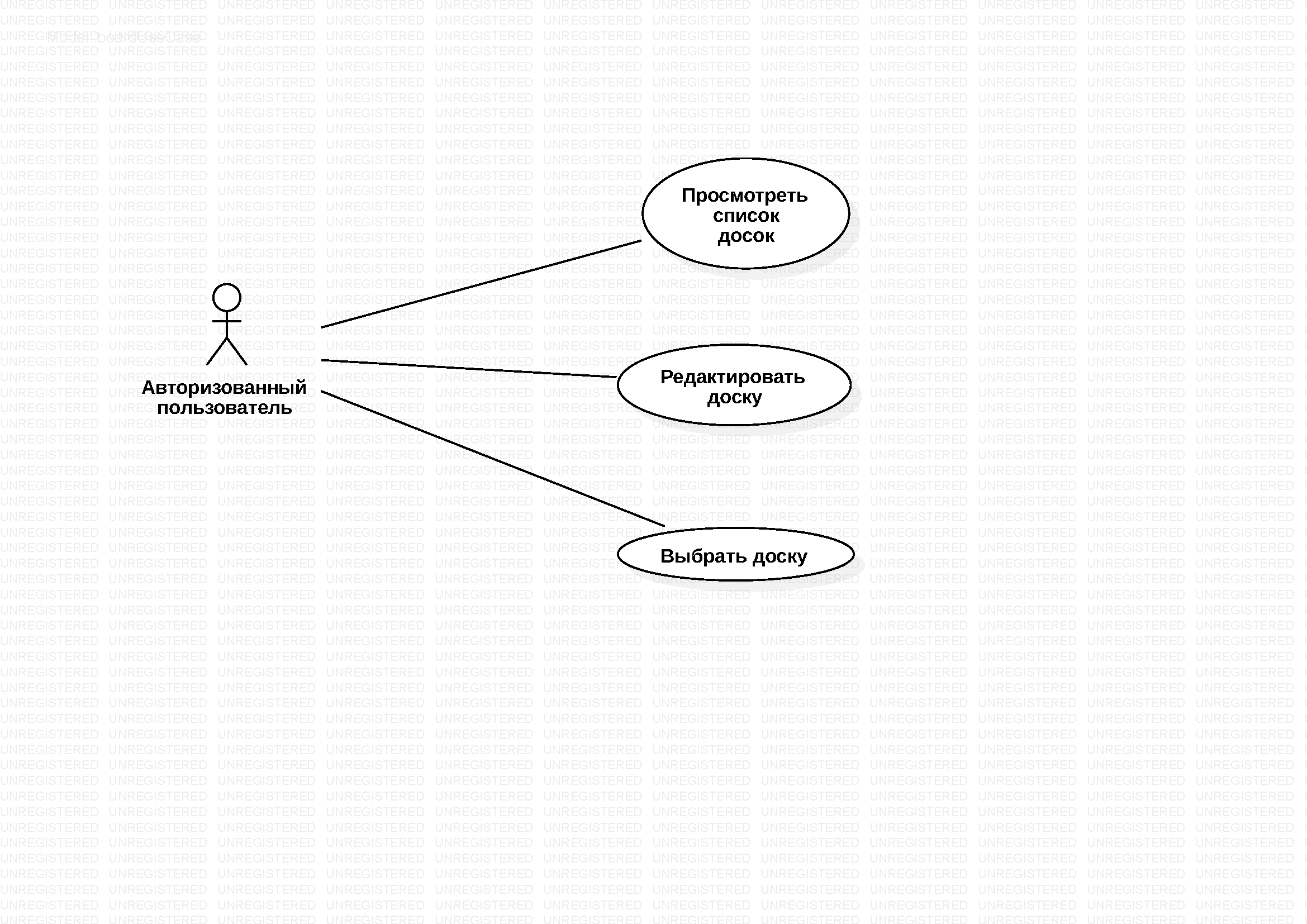
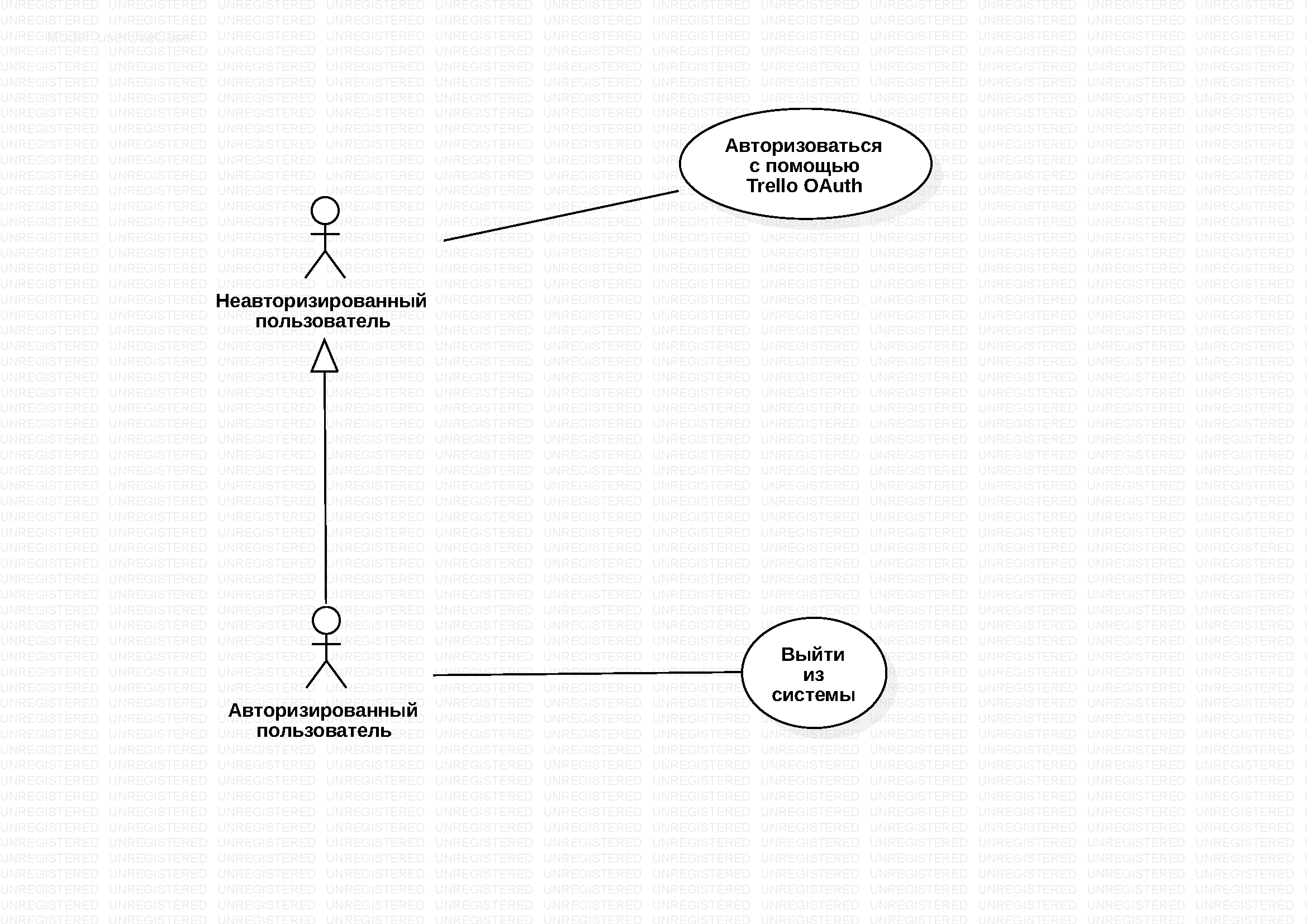
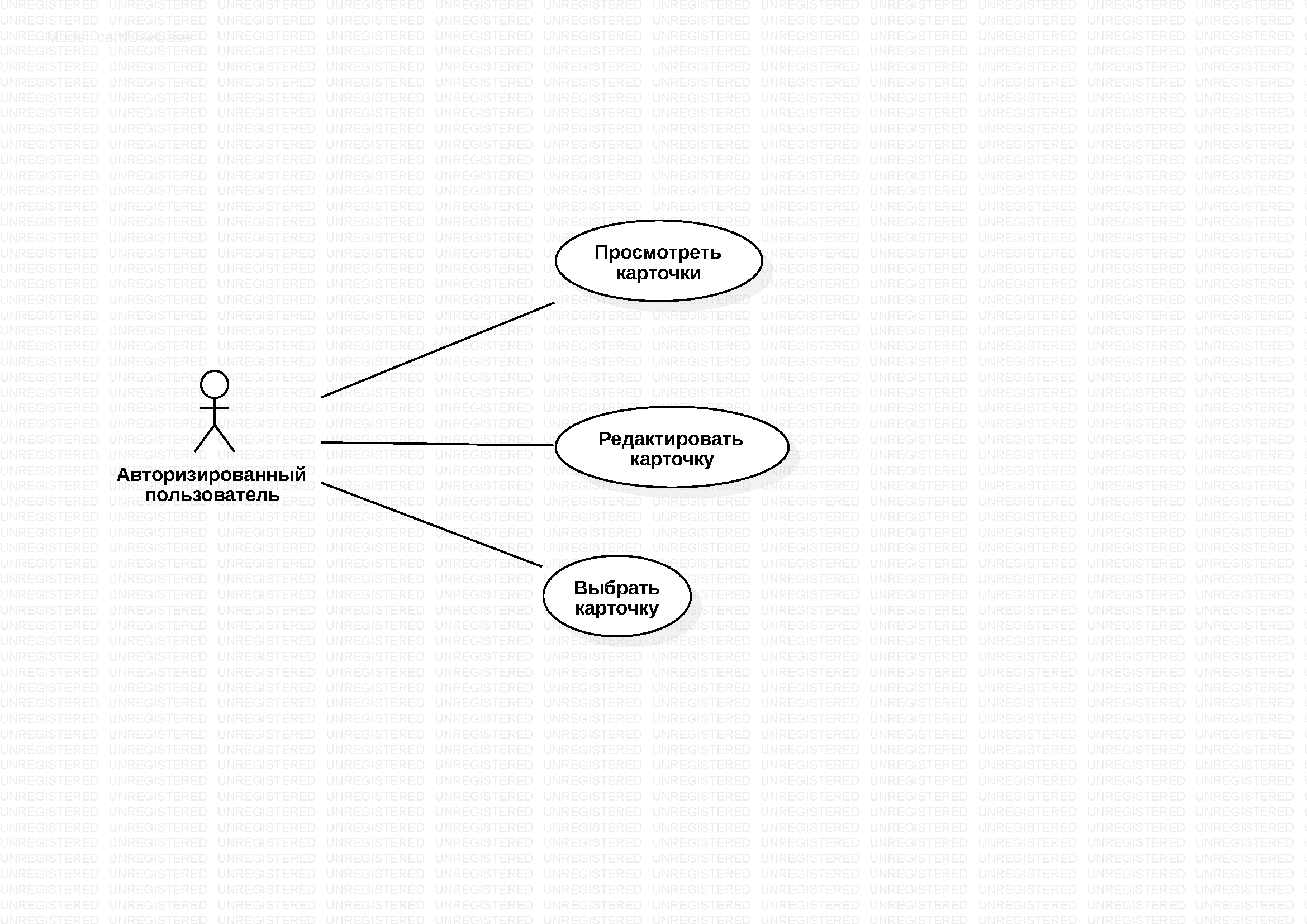
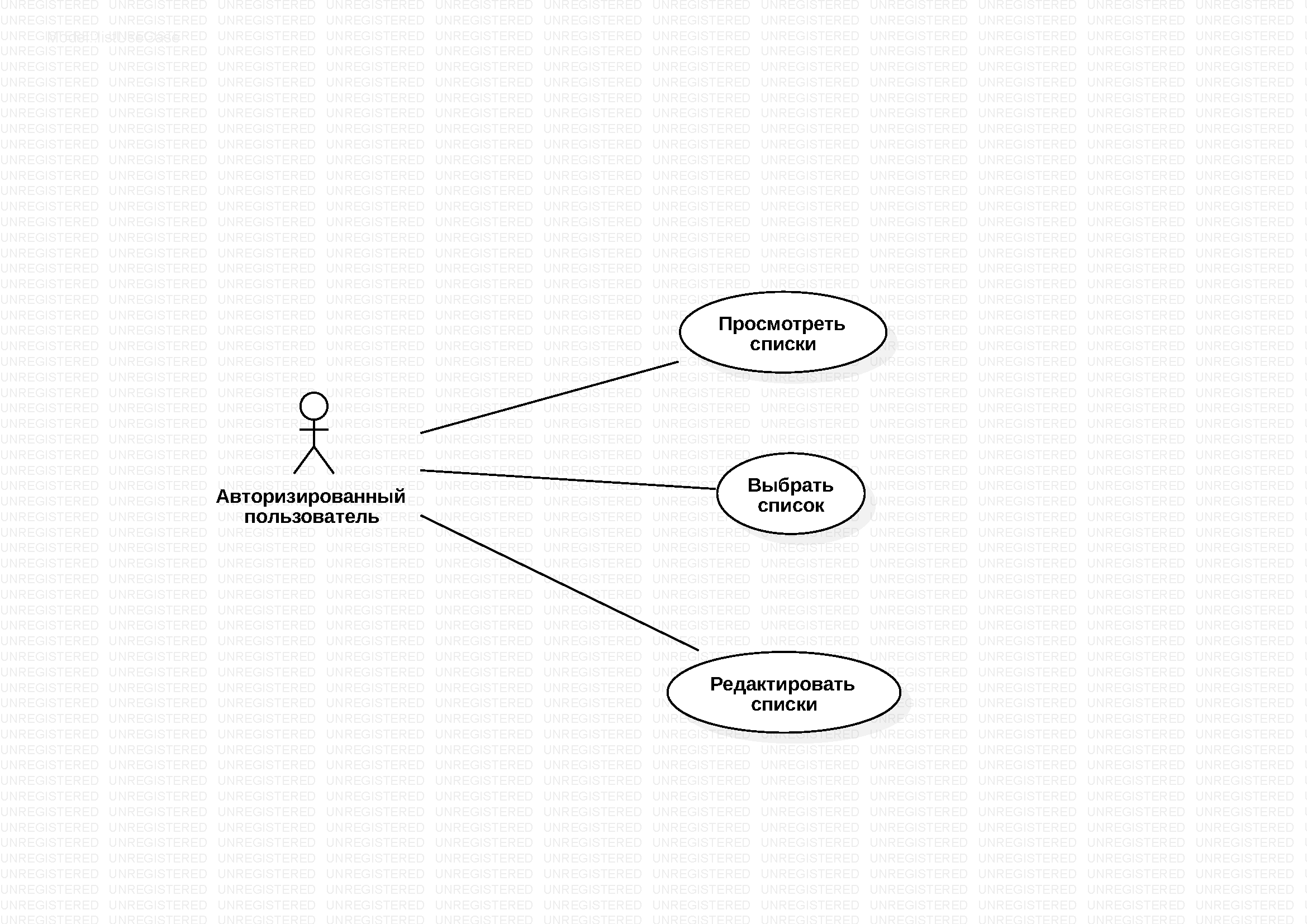
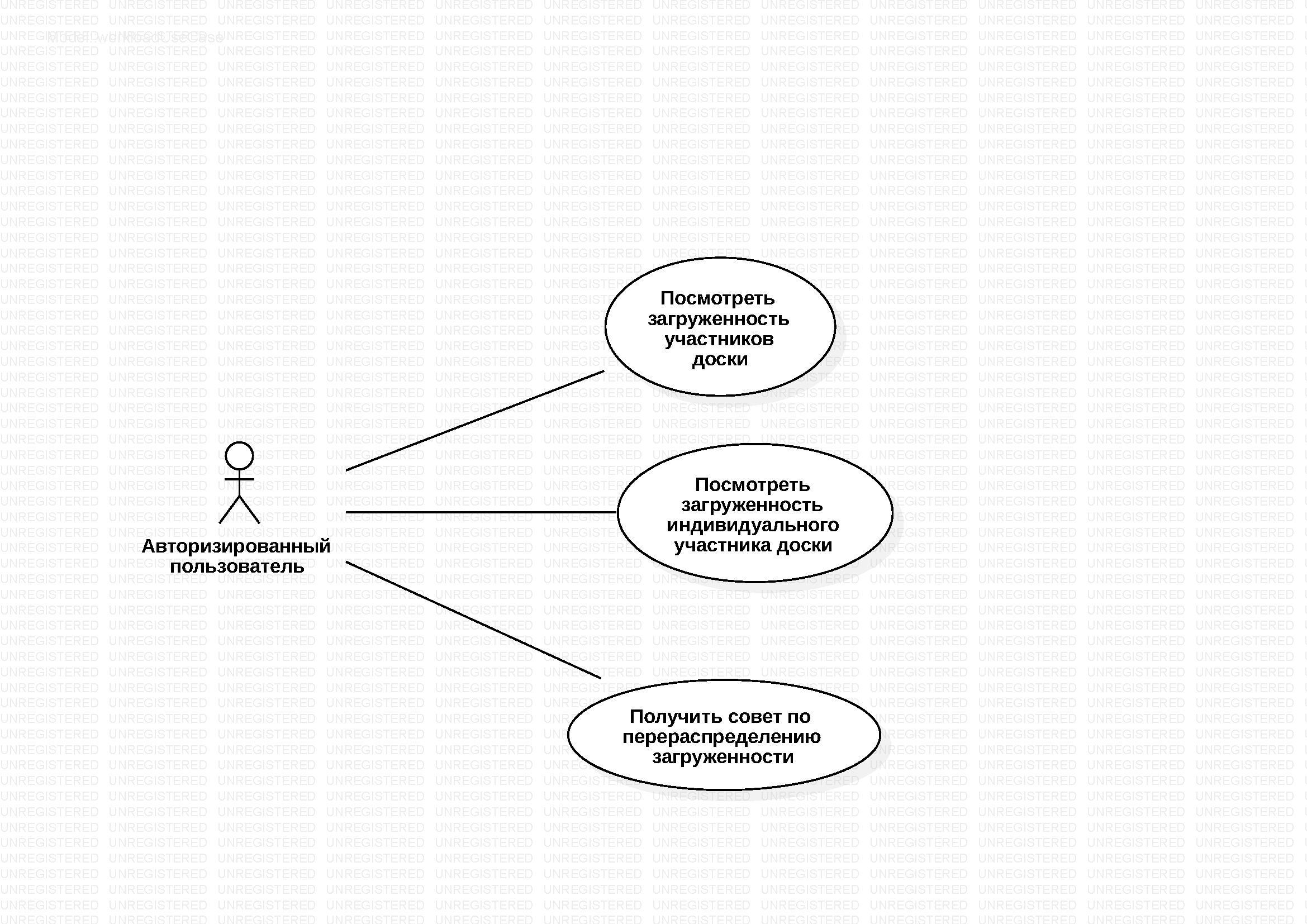


Диаграмма активности



Варианты использования

Для приложения была выбрана монолитная архитектура, так как это обеспечивает большую скорость взаимодействия между элементами приложения. Кроме того, из-за компактности приложения, его нельзя разделить на достаточное число микросервисов чтобы микросервисная архитектура была оправдана. Таким образом, TrelloSuggest состоит из клиентской и серверной частей, а также активно пользуется Trello API. Взаимодействие этих частей приложения показаны на диаграммах коммуникации и взаимодействия:

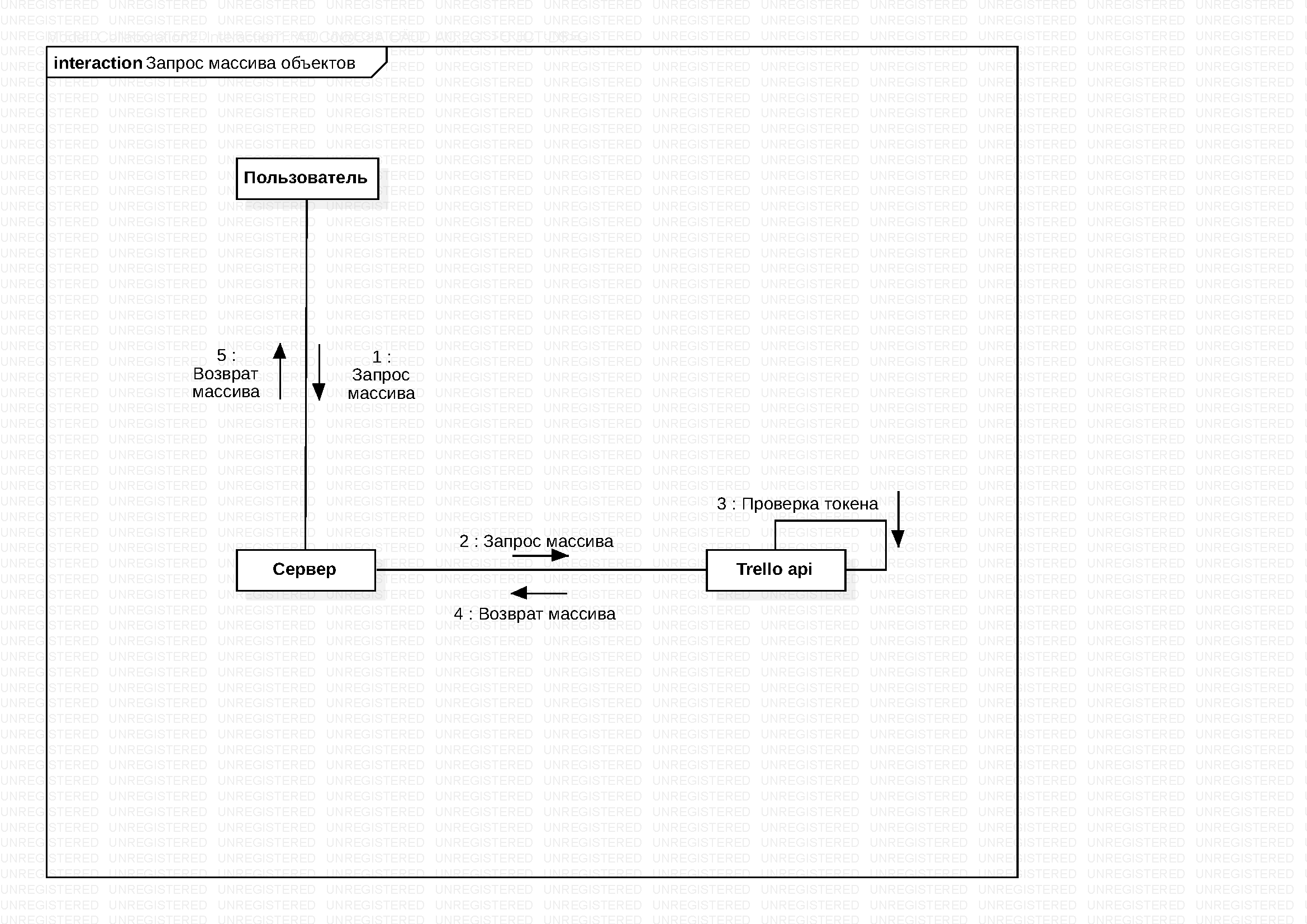
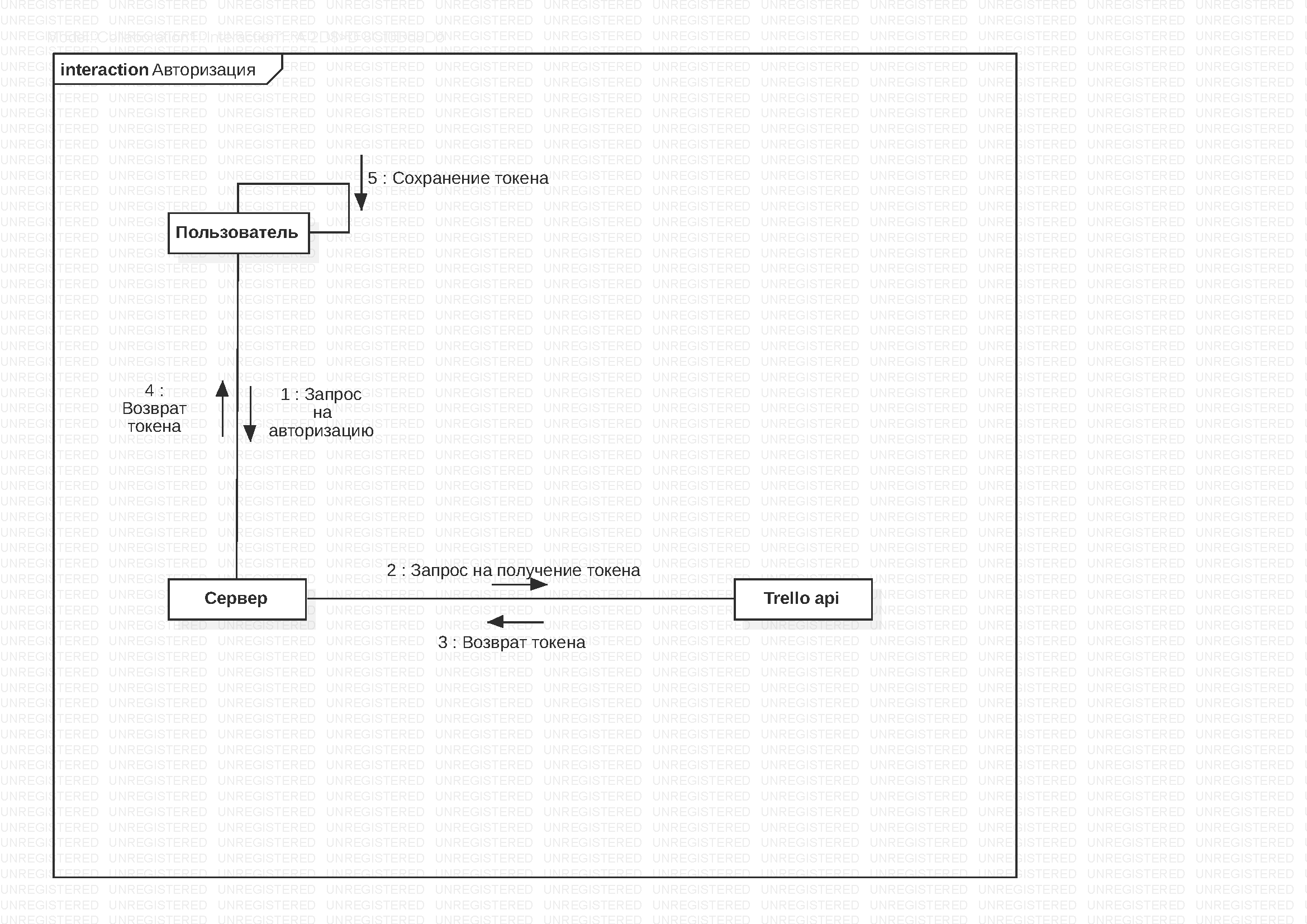
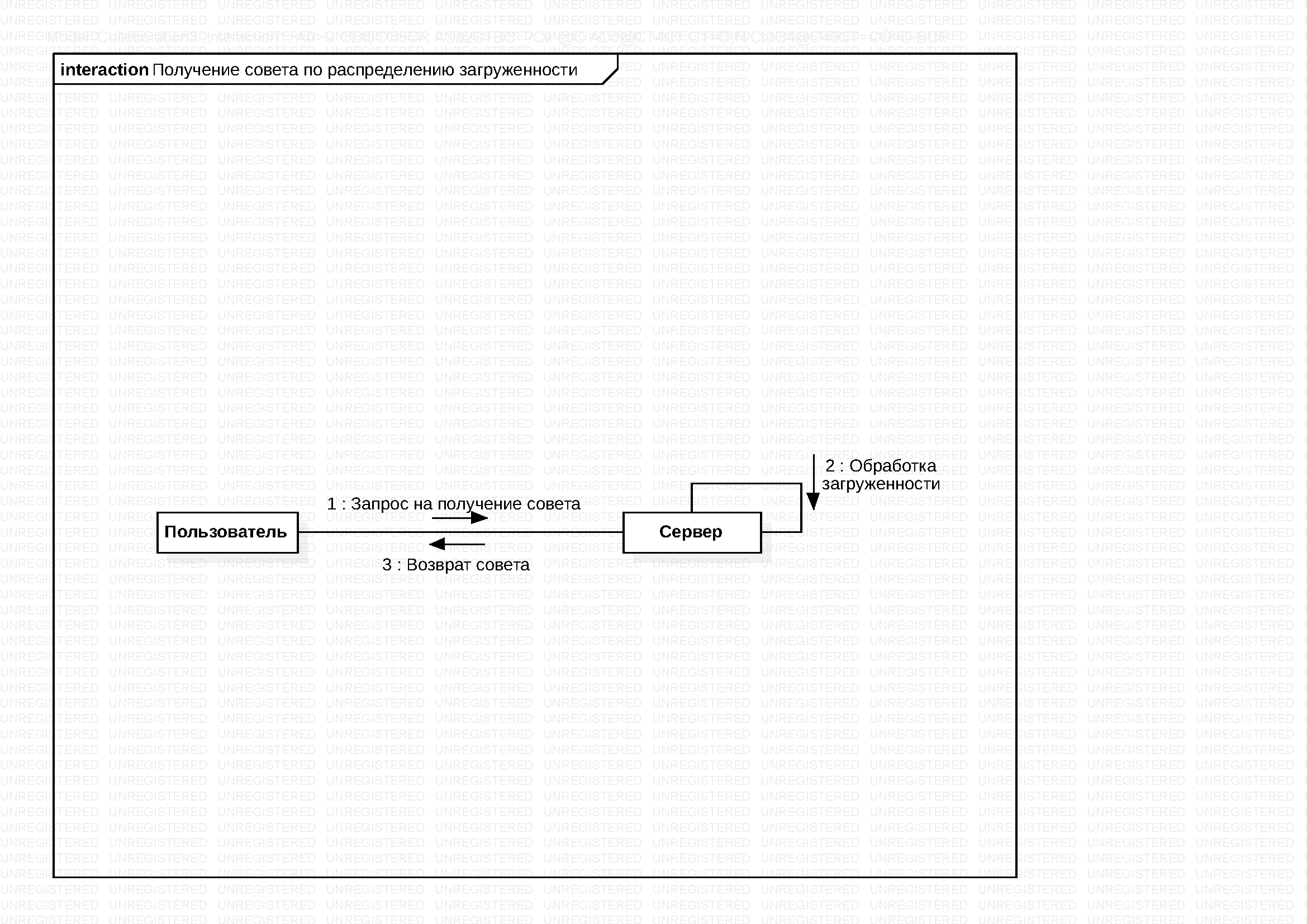


Диаграмма коммуникации

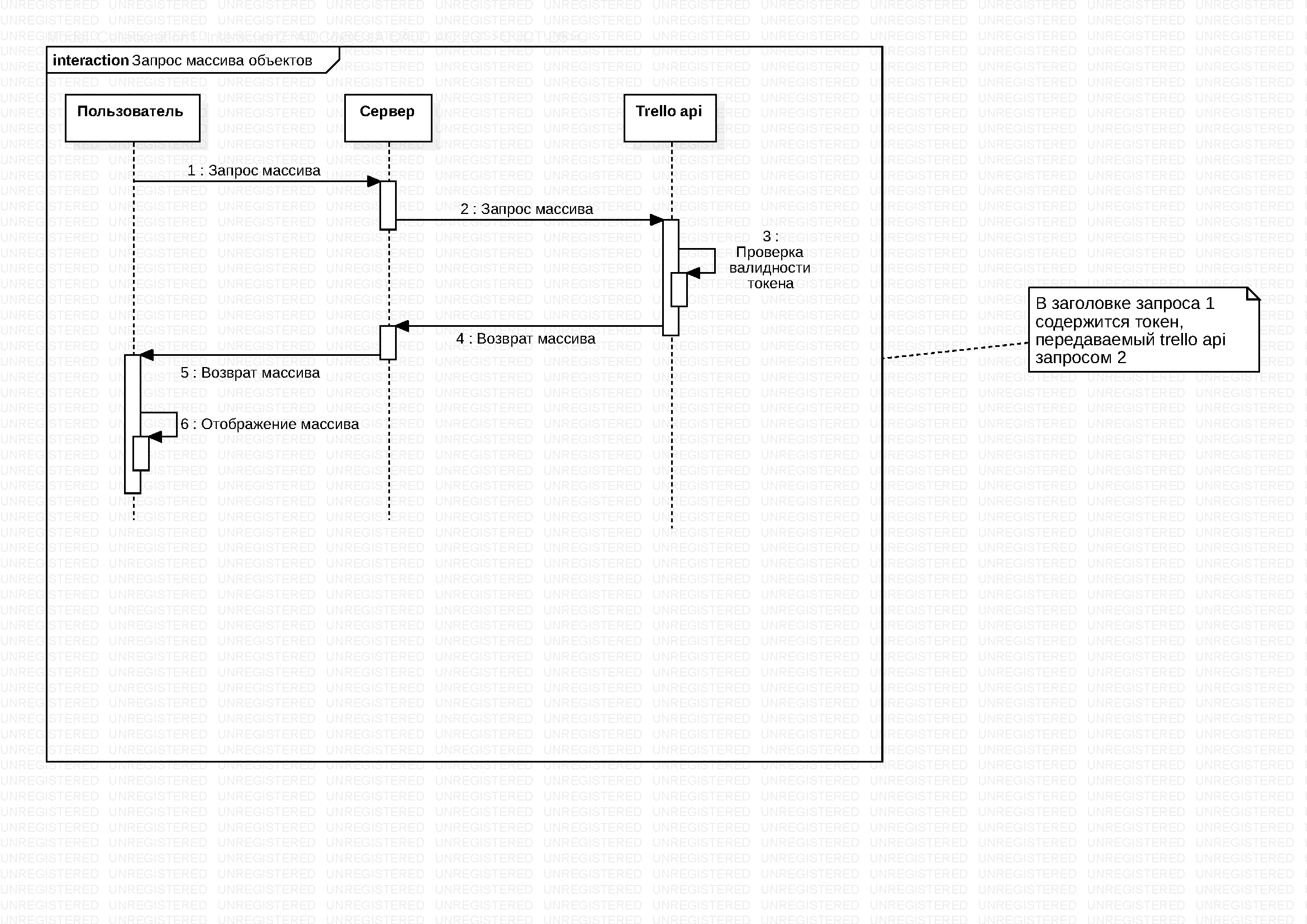
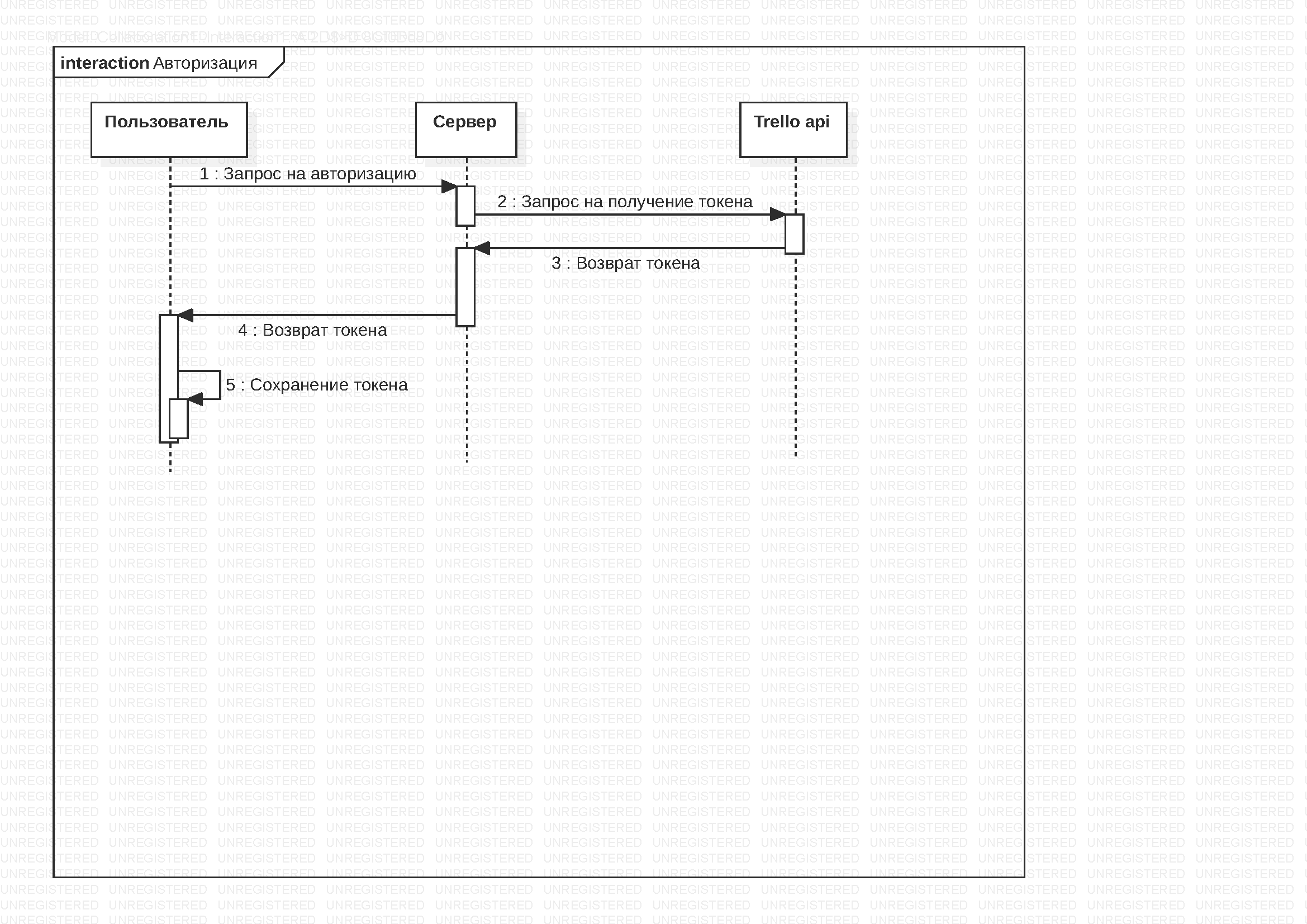
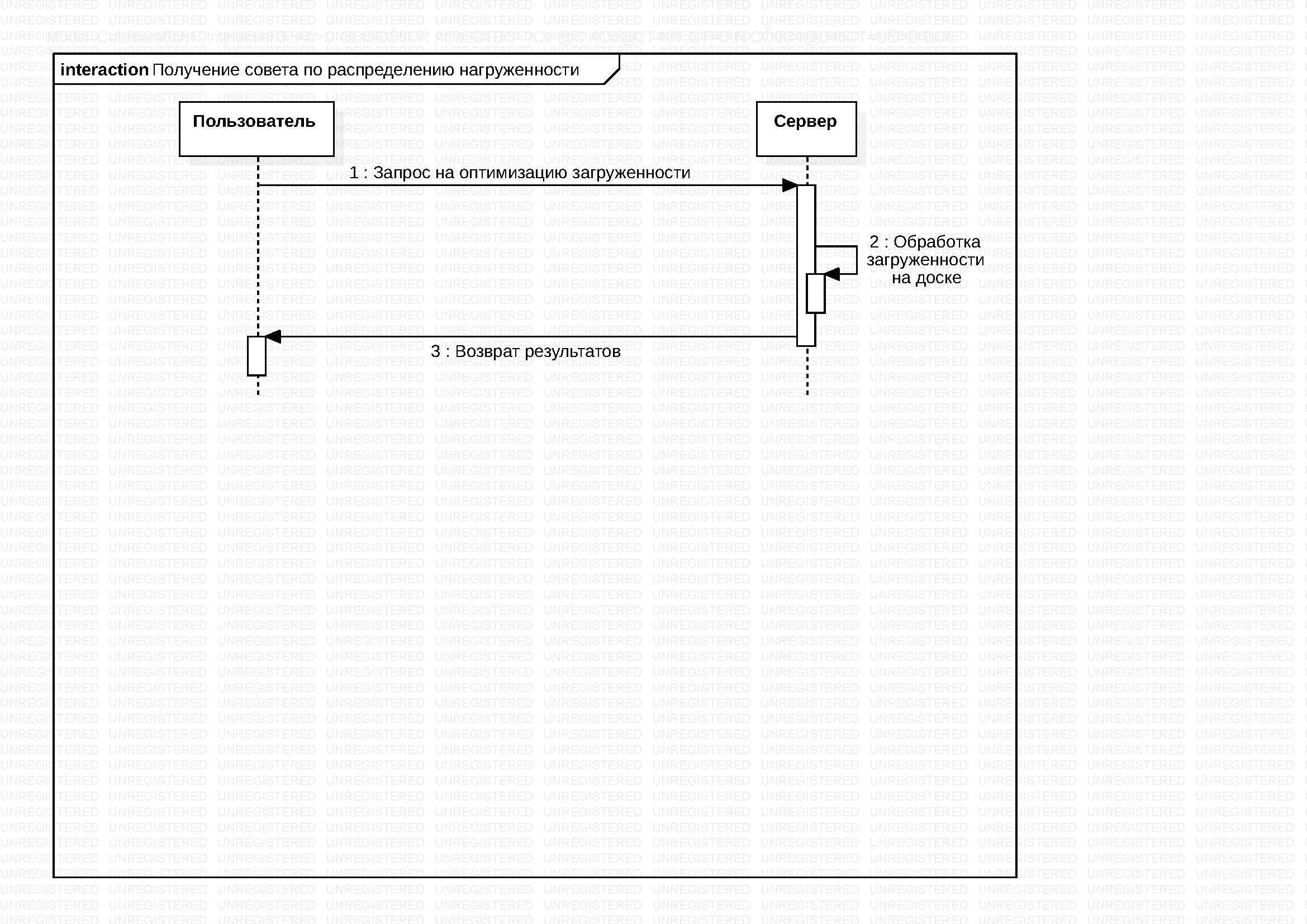


Диаграмма взаимодействия

Так как архитектура была выбрана монолитная, то и вариант развертки был выбран «все на одной машине», хотя в перспективе, при расширении клиентской базы, база данных может быть вынесена на другую машину. На данный момент диаграмма развертывания выглядит так:

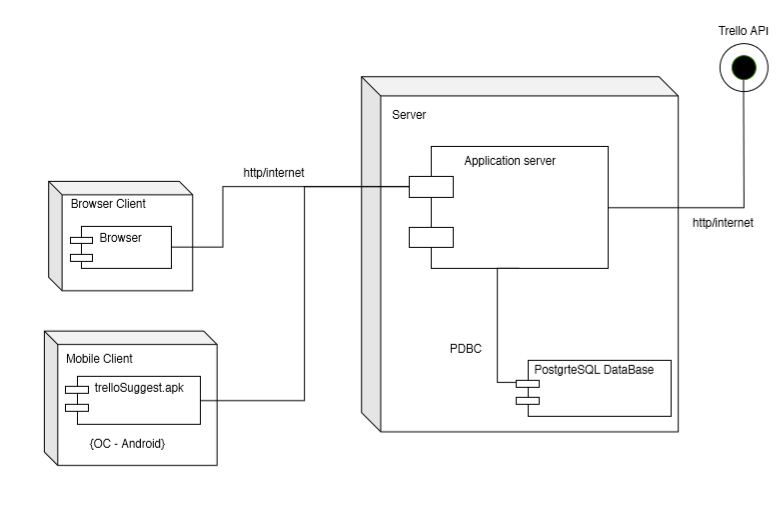


Диаграмма развертывания

Классовая модель была построена на основе паттерна Model-View-Controller. Классы типа Controller (BoardController, MemberController, ListController, CardController) отвечают за работу с данными, которую предоставляют классы типа Model (Card, List, Member, Board).

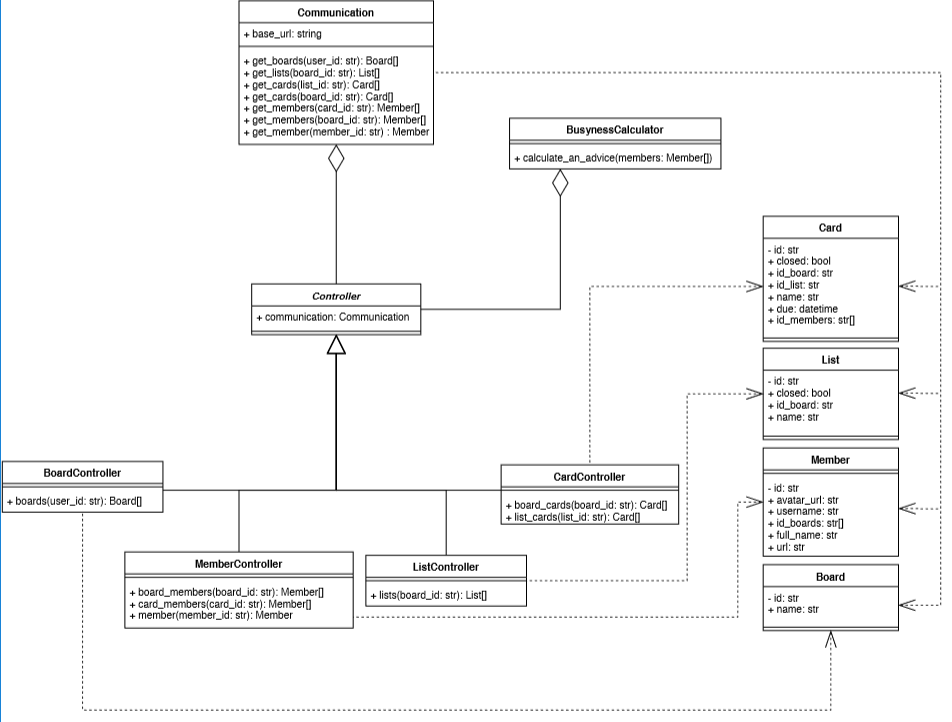


Диаграмма классов

На диаграмме объектов показаны экземпляры классов типа Controller, чтобы показать их конкретное поведение. Каждый из них содержит один и тот же экземпляр класса Communication (паттерн Standalone)

