**Содержание**

[Глава 1 Обзор методов разделения прав доступа в информационных системах 2](#_Toc72786105)

[1.1 Дискреционная модель. 4](#_Toc72786106)

[1.2 Ролевая модель разделения доступа 6](#_Toc72786107)

[1.3 Мандатная модель разделения доступа 8](#_Toc72786108)

[Глава 2 Разработка алгоритма разделения прав доступа 10](#_Toc72786109)

[2.1 Определение ролей 10](#_Toc72786110)

[2.2 Определение прав доступа для обмена сообщениями 12](#_Toc72786111)

[2.3 Алгоритм назначения ролей в группе 12](#_Toc72786112)

[2.4 Алгоритм разделения прав доступа в чате 15](#_Toc72786113)

[Глава 3 Программная реализация системы обмена сообщениями с возможностью разделения прав доступа для пользователей 17](#_Toc72786114)

[3.1 Анализ предметной области 17](#_Toc72786115)

[3.2 Проектирование ПО и выбор архитектуры 18](#_Toc72786116)

[3.2.1 Алгоритм авторизации. 18](#_Toc72786117)

[3.2.2 Алгоритм создания групп 19](#_Toc72786118)

[3.2.3 Алгоритм создания чатов 20](#_Toc72786119)

[3.2.4 Алгоритм отправки сообщений 21](#_Toc72786120)

[3.3 Программная реализация 22](#_Toc72786121)

[3.3.1 Разработка структуры хранения данных 23](#_Toc72786122)

[3.3.2 Разработка серверной части 25](#_Toc72786123)

[3.3.3 Разработка клиентской части 30](#_Toc72786124)

1. Обзор методов разделения прав доступа в информационных системах

Разграничение доступа — это система определения полномочий субъекта в информационной системе и обеспечения их действий строго в рамках установленных для них полномочий. То есть система, которая, с одной стороны, определяет, кому из субъектов разрешён допуск к тем или иным объектам, и с другой стороны, не позволяет им превышать собственные полномочия. Разграничение доступа, как правило, реализуется для решения следующих задач. Во-первых, это обеспечение конфиденциальности информации путём того, что тем субъектам, которые не входят в круг легальных пользователей той или иной конфиденциальной информации, доступ к ней не предоставляется. Для обеспечения целостности информации — здесь можно также добавить и доступность информации — путём того, что пользователям, которые могут нанести вред целостности или доступности информации, доступ к ней не предоставляется, а также все их действия могут таким образом фиксироваться за счёт того, что они действуют в рамках своих прав и перед этим прошли процедуру аутентификации, то есть все действия пользователей записываются и могут, например, быть восстановлены из резервных копий состояний информации, которая стала жертвой каких-то не санкционированных изменений. Разграничение доступа также позволяет обеспечивать контроль действий пользователя за счёт того, что каждый пользователь действует строго в рамках своих полномочий в системе и не может их никаким образом превысить. И разумеется, разграничение доступа служит реализацией принципов минимизации полномочий и разделения обязанностей, то есть каждый из субъектов в системе обладает только тем набором прав, который соответствует его должностным обязанностям, и весьма желательно, чтобы ни к каким дополнительным ресурсам системы пользователь бы доступа не имел, поскольку это является уязвимостью, которой может воспользоваться, если не он сам, то, например, нарушитель, так или иначе обойдя систему аутентификации и представившись тем или иным легальным пользователем. Системы разграничения доступа, как правило, оцениваются по следующим параметрам. Трудоёмкость первоначальной настройки, то есть оформление прав всех пользователей для того, чтобы система могла начать функционировать и корректно предоставлять пользователям права в информационной системе. Возможность настройки полномочий в нетипичных случаях также характеризует качество системы разграничения доступа. Здесь можно привести следующий пример. Допустим, существует два отдела, в которых есть руководство. Должны ли руководители разных отделов иметь доступ к информации, относящейся к ведению другого отдела. Например, должен ли бухгалтер так или иначе иметь доступ к файлу паролей, который, разумеется, относится к ведению службы безопасности? И наоборот, должен ли администратор безопасности информационной системы иметь доступ к файлу с ведомостью на получение заработной платы сотрудников? С одной стороны, у них примерно равный уровень у этих двух лиц, они руководителя отделов, а, соответственно, информация, о которой идёт речь, относится к более низкой, то есть доступной как минимум кому-то из руководства отделов. Но эти перекрёстные возможности доступа, должны ли они быть допущены или должны ли они быть заблокированы, зависит от решения руководства, а возможность реализации на практике таких прав доступа зависит от того, какая система разграничения доступа избрана. Как мы увидим чуть далее, некоторые модели позволяют такие тонкие настройки прав пользователей, а некоторые — нет. И ещё одно важное качество, важное свойство, которое влияет на то, насколько система разграничения доступа удобна или нет, это удобство добавления нового субъекта или объекта. Под субъектом здесь подразумевается пользователь информационной системы, а под объектом — файл или иной информационный ресурс. Практически любая система разграничения доступа не лишена следующих уязвимостей, просто связанных с её идеологией, не потому, что она плоха или хороша, а потому, что это система, которая действует именно так. Во-первых, после того, как субъект благополучно пройдёт процедуру авторизации, то есть получит доступ к тем полномочиям, которые предоставлены ему в системе, контроль его подлинности не осуществляется, как правило, такая процедура проходится единожды. Хотя в особо критичных ситуациях в критичных операциях может потребоваться дополнительное подтверждение подлинности субъекта, и это является хорошим решением. И наконец, система разграничения доступа сама по себе, без поддержки другими средствами защиты информации, никаким образом не ограничивает действия привилегированных пользователей, то есть администраторов безопасности и тех сотрудников, например, службы безопасности, которые производят настройку системы разграничения доступа, которые имеют полномочия предоставлять права другим пользователям, например, устанавливать для них пароли и подобные административные действия реализовывать. Разграничение доступа, как правило, строится на одном из трёх следующих принципиальных подходов. Существуют и некие другие подходы, которые являются, например, частными случаями описанных либо какими-то их сочетаниями, но выделяют три классические модели. Это дискреционное разграничение доступа, мандатное разграничение доступа и ролевое разграничение доступа, о которых мы далее и поговорим подробнее

## Дискреционная модель.

Она основывается на разграничении доступа с использованием поименованных субъектов и объектов на основе установленных прав доступа для каждой пары. Таким образом, дискреционное разграничение доступа основывается, по сути, на формировании так называемой матрицы прав доступа, то есть такой формы представления реализации дискреционного разграничения доступа в виде матрицы. Строки и столбцы данной матрицы соответствуют субъектам и объектам, а на пересечении этих заголовков строк и столбцов находятся права, которые данный субъект имеет по отношению к данному объекту. Например, только право чтения или права на чтение и запись, либо какие-то другие наборы прав, либо в ряде случаев, если речь идет только о возможности доступа к объекту, просто признак наличия или отсутствия права доступа, например, единица или ноль. В этом случае система разграничения доступа действует по следующей схеме. Каждый из пользователей обращается к системе разграничения доступа через сервер авторизации для того, чтобы осуществить попытку доступа к тому или иному объекту в информационной системе. Сервис авторизации, который, собственно, принимает решение о том, должен ли тот или иной пользователь быть авторизован для такого действия, обращается к базе данных, в которой хранится матрица доступа. Эта матрица доступа, структура которой была выше описана, содержит строку, в которой указано уникальное имя данного пользователя, то есть его идентификатор, и в одном из столбцов — название того объекта, к которому пользователь желает обратиться. Если на пересечении этой строки и этого столбца находится информация о том, что данный пользователь действительно авторизован для реализации такого доступа, то доступ предоставляется, в противном случае в доступе отказывается — так работает эта модель. Данная модель имеет следующие достоинства. В ней возможна индивидуальная настройка прав для каждого пользователя, поскольку заполняется полная матрица соответствия субъектов и объектов. Таким образом, в модели с двумя начальниками двух отделов можно совершенно спокойно настроить для каждого из них нужный уровень доступа к тем или иным файлам, относящимся к ведению их отделов, и не предоставлять им прав на доступ к файлам других отделов, несмотря на то, что иерархически эти пользователи более привилегированны, чем, например, пользователи, которые имеют право к этим файлам, может быть, в некотором роде являются их начальниками, но по роду деятельности не должны иметь права для доступа к этим файлам, и можно такой запрет реализовать. Для каждого объекта можно настроить права доступа, то есть ту группу субъектов, которая будет иметь к ним доступ. И нет никакой зависимости между сходством в уровне секретности объектов или в должностном уровне субъектов. Два системных администратора, например, могут иметь совершенно разный набор прав доступа к объектам, например, на основе их опыта, на основе того, чем конкретно они занимаются в организации, на каких, допустим, неисправностях специализируются или на обеспечении каких сервисов они специализируются. Не лишена данная модель и недостатков, которыми являются следующие обстоятельства. Во-первых, для реализации, то есть для первоначального внедрения такой модели требуется полностью заполнить матрицу субъектов и объектов в каждой ее клеточке, по сути, на пересечении каждого столбца и каждой строки установить конкретные права доступа данного субъекта к данному объекту. Из этого следует, что для добавления нового субъекта или объекта требуется заполнить все элементы соответствующей строки или столбца матрицы доступа. То есть если мы добавляем нового субъекта, то появляется строка, в которой нужно прописать права его доступа ко всем существующим объектам, а если новый объект — то, соответственно, для всех субъектов, которые у нас описаны в данной системы, чьи идентификаторы существуют, требуется установить их права доступа к новому объекту.

## Ролевая модель разделения доступа

Ролевая модель, как следует из ее названия, основывается на так называемых ролях субъектов информационной системы. По сути, она является некоей оптимизацией дискреционного разграничения доступа. Под ролью в данной модели разграничения доступа подразумевается совокупность прав доступа субъекта к объектам информационной системы. То есть существует некое эталонное значение этих прав доступа субъекта к объекту, которое единожды сохраняется и именуется каким-то названием. Это название далее называется ролью. О том, что это такое, сейчас на примере будет более понятно. Например, может быть роль «системный администратор». В этом случае существует некий набор прав, который является как бы стандартным для системного администратора. Может быть роль начальника отдела, роль оператора, роль бухгалтера. И для каждой такой роли прописывается, как в дискреционной системе разграничения доступа прописывается набор прав доступа к различным объектам системы. В этом случае достигается ряд достоинств. Во-первых, добавление нового субъекта не требует заполнения строки матрицы при наличии подходящей роли. В самом деле, для того чтобы добавить нового сотрудника, допустим, недавно принятого на работу в организацию в информационную систему, по сравнению с дискреционной моделью разграничения доступа уже не требуется полностью заполнять всю строчку его прав доступа ко всем объектам в информационной системе, которых может быть, допустим, несколько сотен. При этом количество ролей ничем не ограничено, и если требуются создать какие-то особенные роли, например не просто начальника отдела, а начальника какого-то конкретного отдела, и помощника начальника какого-то конкретного отдела или еще кто-то, то это ничто не мешает сделать. И таким образом можно создать адекватную роль для каждого сотрудника: с одной стороны, для тех сотрудников, которые является, скажем, уникальными — каких-то руководителей, а с другой стороны, для сотрудников, которые выполняют какие-то типовые операции, например операторов информационной системы, системных администраторов и прочих специалистов, можно использовать уже готовые подходящие роли. При этом роль не предполагает полного нисходящего предоставления доступа к объектами информационной системы, то есть свободна от того недостатка, который есть в мандатной модели разграничения доступа. Можно какому-то конкретному администратору, например, предоставить доступ к файлу с уровнем секретности, который в мандатной модели имел бы метку конфиденциальности «секретно», но при этом относится, допустим, к вопросам администрирования информационной системы. И при этом не предоставить ему права доступа к тем объектам, которые в мандатной модели имели бы более низкие метки конфиденциальности. Роль вполне это допускает. Полное нисходящее предоставление доступа к объектам не предполагается. При этом есть и свои недостатки у ролевой модели разграничения доступа. Если доступ к объектам основывается на отдельных объектах, а не группах, то добавление нового объекта аналогично дискреционной модели. В классической ролевой модели о группировании объектов речи не идет. Создаются роли для субъектов, а объекты, в общем-то, описываются независимо отдельными строками в этой матрице. По сути, речь идет о формировании такой матрицы, где каждой строкой является не субъект, а роль. Если под одну роль подходит более одного субъекта, то случается некая оптимизация. Если же — и это как раз второй ее недостаток — для каждого субъекта требуется отдельная роль — все они такие незаменимые, непохожие друг на друга, совершенно уникальные, — такое часто случается в небольших организациях, где каждый сотрудник выполняет ряд функций, практически не имея никакой замены, никакой помощи со стороны других пользователей. Просто потому, что есть, допустим, один системный администратор, один бухгалтер, один генеральный директор, один, допустим, менеджер по продажам, если, к примеру, речь идет о, допустим, турагентстве, и у каждого из них есть свой уникальный набор прав в системе. В этом случае под каждого из них приходится создавать отдельную роль. В этом случае ролевое разграничение доступа сводится к дискреционной модели, то есть никакой оптимизации не происходит. Мы с вами поговорили сегодня о различных способах контроля и разграничения доступа субъектов в систему. И далее в следующих лекциях речь пойдет о других категориях методов защиты информации, организационные меры защиты информации, к которым относятся и разграничения доступа, могут быть поддержаны и усилены.

## Мандатная модель разделения доступа

Мандатная модель основана на использовании так называемых меток конфиденциальности объектов и уровней допуска субъектов. Под метками конфиденциальности подразумевается присваиваемый объекту информационной системы уровень конфиденциальности, устанавливающий его место в иерархии уровней доступности объектов. Например, в организации могут быть установлены четыре типа меток конфиденциальности — несекретная информация, информация только для служебного пользования, секретная и совершенно секретная информация, обозначенные соответствующими шифрами. Это означает, что вся информация в системе как бы иерархически структурируется на четыре уровня доступности. Это есть метка конфиденциальности, принадлежность того или иного объекта к одному из этих уровней. Соответственно, для субъектов вводится понятие уровня допуска. Это такой уровень конфиденциальности, к объектам с метками конфеденциальности не выше которого субъект имеет доступ. Например, если субъект имеет уровень доступа ДСП, то есть для служебного пользования, то он получает доступ ко всем объектам с метками секретности «не секретно» и «для служебного пользования», но при этом не имеет доступа ни к каким объектам с иными метками. Это означает, что в моделе в примере с двумя начальниками каждый из них получит одинаковый уровень доступа ко всем объектам системы, поскольку, скорее всего, они получат одинаковый уровень допуска, а значит, множество тех объектов, к которым они фактически будут иметь доступ, будет абсолютно одинаковым. Такая система действует по следующей схеме. Здесь, для того чтобы определить, может ли тот или иной пользователь получить доступ к тому или иному объекту, сравнивается уровень допуска пользователя и метка конфиденциальности объекта. Если уровень допуска не уступает метке конфиденциальности или превосходит её, то доступ предоставляется. В рассмотренном примере пользователь с уровнем допуска «секретно» пытается обратиться к объекту с меткой конфиденциальности «для служебного пользования». Соответственно, ему доступ предоставляется. А пользователь, обладающий более низким, чем «для служебного пользования», уровнем допуска, получает отказ в доступе к данном объекту. Данная система имеет следующие достоинства. Для реализации её достаточно установить кажому объекту метку конфиденциальности, а каждому субъекту — уровень допуска. Соответственно, это делается значительно быстрее и проще, чем полное заполнение матрицы для каждой пары субъект-объект. Из этого следует, что добавление нового объекта или нового субъекта не требует дополнительного редактирования прав других субъектов или объектов. Если в системе появляется новый субъект, ему устанавливается уровень допуска — и система может функционировать далее. Если появляется новый объект, ему приписывается метка конфиденциальности — и снова никаких других действий в системе не требуется. Из этих достоинств, в свою очередь, следуют и недостатки. Доступ к объектам одного уровня секретности предоставляется или ограничивается совместно. Нет возможности разделить всё множество объектов с уровнем секретности, например, «для служебного пользователя» на подмножество, доступное одному начальнику отдела, и подмножество, ему недоступное. И вторая негативная черта данной модели разграничения доступа — это отсутствие возможности разграничения доступа в пределах одного уровня допуска или с исключениями на более низких уровнях доступа. Нельзя предоставить какому-либо субъекту доступ к избранному подмножеству всех объектов с меткой конфиденциальности уровня «секретно», а часть, допустим, несекретных документов или документов «для служебного пользования» сделать для них недоступными просто потому, что, например, эти документы не относятся к их сфере деятельности, а касаются совершенно других вопросов. Исходя из принципа минимизации полномочий, это следовало бы сделать, но мандатная модель разграничения доступа этого не позволяет.

1. Разработка алгоритма разделения прав доступа

## Определение ролей

При создании группы создатель приглашает нескольких пользователей в новую группу. При этом, с момента создания за пользователями закрепляются роли: за создателем группы закрепляется роль «Владелец», а за остальными пользователями – роль «Участник». В уже созданной группе Владелец может изменять роли для участников и назначить им либо роль под названием «Модератор» либо роль под названием «Администратор». Администратор и Модератор так же могут приглашать пользователей для участия в группе, однако Владелец и Администратор могут приглашать пользователей определяя их как обычных «Участников» либо, как «Посетителей». Модератор может приглашать новых участников, определяя их в группе только как «Посетителей». Роль посетитель обладает наименьшим количеством прав, пользователи, имеющие данную роль, способны только посещать те чаты группы, которые им это позволяют, и не могут посещать чаты, которые обладают обычной настройкой прав доступа. Данная роль существует для того, чтобы Модераторы могли приглашать в группу новых пользователей – кандидатов на получение другой роли или на удаление из группы, что может осуществить либо «Администратор» либо «Владелец», тем самым, образуя так называемый «Список ожидания». Каждая роль обладает определенным списком прав.

Права, доступные пользователю, имеющему роль «Владелец»:

* Назначать пользователям группы роли: «Администратор», «Модератор», «Участник», «Посетитель»,
* Добавлять и удалять пользователей,
* Создавать и удалять чаты,
* Изменять права доступа участников в чатах,
* Обладать всеми правами в чатах,
* Изменять название группы и названия чатов,

Права, доступные пользователю, имеющему роль «Администратор»:

* Назначать пользователям группы роли: «Модератор», «Участник», «Посетитель»,
* Добавлять и удалять пользователей,
* Создавать и удалять чаты,
* Изменять права доступа участников в чатах,
* Обладать всеми правами в чатах,
* Изменять названия чатов,

Права, доступные пользователю, имеющему роль «Модератор»:

* Добавлять пользователей, назначая им роль «Посетитель»
* Удалять посетителей, которых назначил он сам,
* Создавать чаты,
* Удалять только те чаты, которые создал он сам
* Изменять права доступа участников, имеющих роли «Участник» или «Посетитель», в чатах, которые создал он сам
* Обладать всеми правами в чатах, которые создал он сам
* Изменять названия чатов, которые создал он сам

Права, доступные пользователю, имеющему роль «Участник»:

* Посещать чаты со стандартным набором ограничений

Права, доступные пользователю, имеющему роль «Посетитель»:

* Посещать чаты, в которых разрешено посещение пользователями, имеющими данную роль.

## Определение прав доступа для обмена сообщениями

Для обмена сообщения в группе нужно использовать чат. Чатов может быть несколько в одной группе, в каждом чате могут быть определены права на его использование любым пользователем группы. Существует несколько, заранее определенных правил поведения пользователя в чате (такие как, чтение сообщений, отправка сообщений и т.д.) Изначально, при создании чата, для каждого права в новом чате определяется список пользователей, которые будут обладать данным правом в чате. Право может быть как и обычным - включающим в свой список всех пользователей не ниже уровня «Участника», или определяющим привилегию по данному праву конкретных участников, которые имеют роль, предполагающую получить привилегию. В последующем, можно будет определить тип списка. Если тип списка будет определен как «Белый», то это будет означать, что все пользователи, которые записаны в этот список будут обладать тем правом, в чей список они записаны. Если же тип списка будет определен как «Черный», то это будет означать, что все пользователи, которые записаны в этот список не будут обладать тем правом, в чей список они записаны.

## Алгоритм назначения ролей в группе

Как было описано ранее, создатель группы имеет роль «Владелец», а все другие участники группы имеют роль «Участник». Роль пользователя определяется при переходе по ссылке на группу. При переходе по ссылке, браузер передает на сервер идентификатор пользователя и идентификатор группы, по которым, сервер находит в базе данных запись, которая определяет роль пользователя в группе. Сервер, ответом на запрос браузера, высылает браузеру информацию о том, какую роль имеет пользователь, и, на основе этой информации браузер выводит на экран те элементы интерфейса, которые разрешено видеть пользователям данной роли. Схема алгоритма представлена на рисунке 2.1.

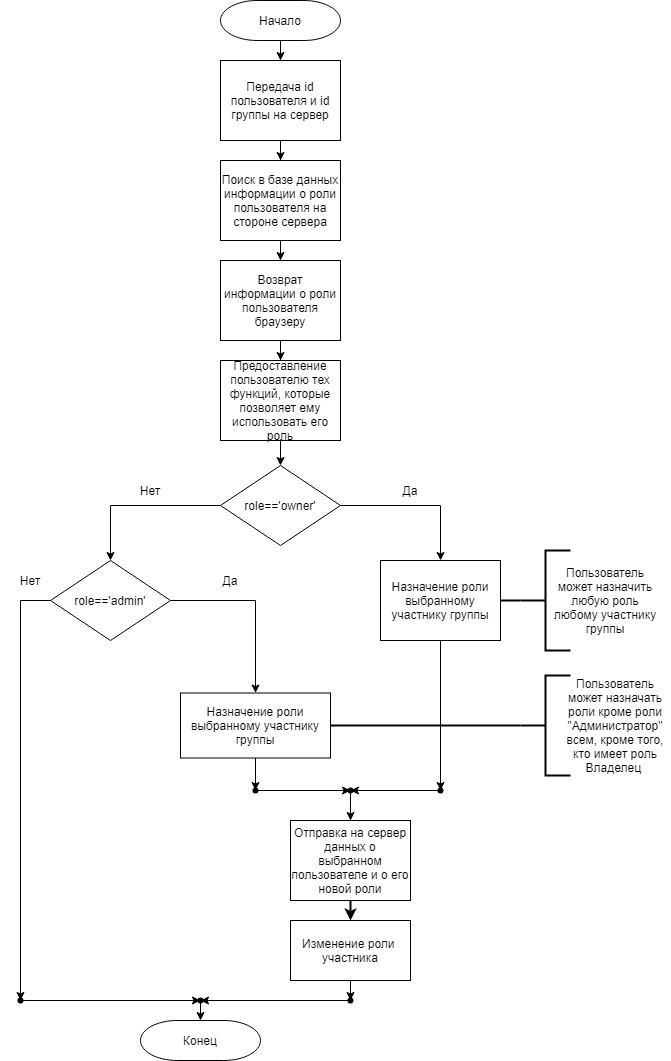


Рисунок 2.1 – Схема алгоритма назначения ролей в группе

## Алгоритм разделения прав доступа в чате

Права пользователя определяется при переходе по ссылке на группу. При переходе по ссылке, браузер передает на сервер идентификатор пользователя и идентификатор группы, по которым, сервер находит в базе данных запись, которая определяет права доступа пользователя во всех чатах в выбранной группе. Сервер, ответом на запрос браузера, высылает браузеру информацию о том, какую роль имеет пользователь, и, на основе этой информации браузер выводит на экран те элементы интерфейса, которые разрешены пользователю полученной информацией о его правах. Схема алгоритма представлена на рисунке 2.2.

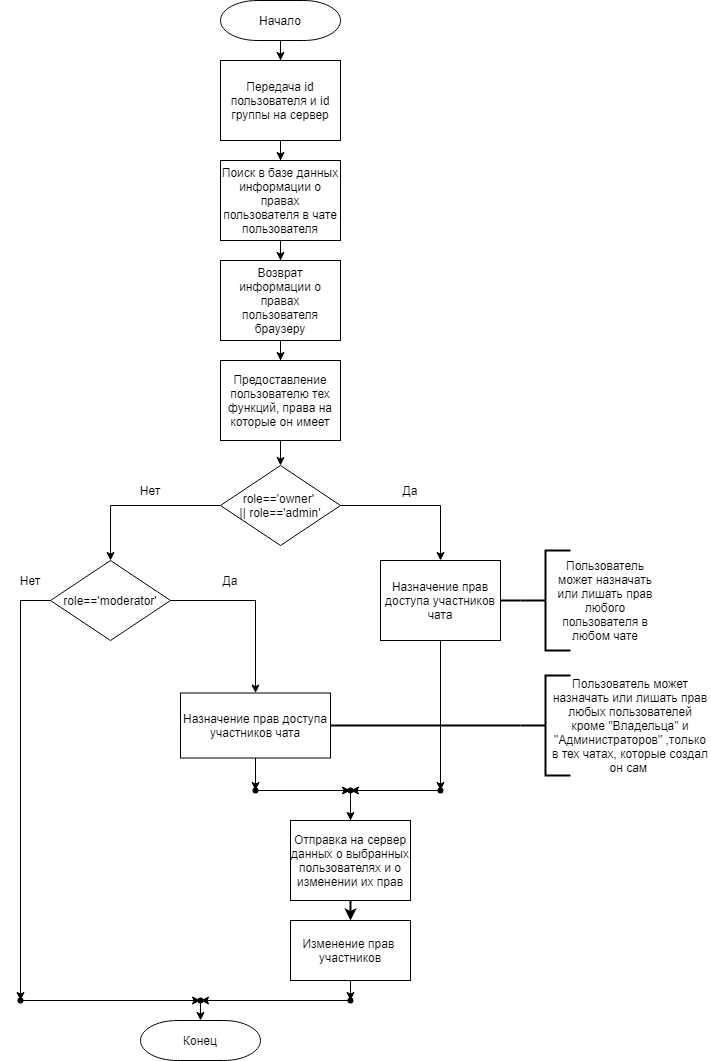


Рисунок 2.2 – Схема алгоритма разделения прав доступа в чате

1. Программная реализация системы обмена сообщениями с возможностью разделения прав доступа для пользователей

## Анализ предметной области

В качестве программного обеспечения была разработана система обмена сообщениями (мессенджер), основу работы которого определяют взаимодействия следующих сущностей:

* Пользователи;
* Группы;
* Чаты;
* Сообщения;
* Роли пользователей в группах;
* Права пользователей в чатах;

Первая сущность определяет пользователя в системе, и является уникальной для каждого пользователя. Каждый пользователь может принимать участие в нескольких группах, при том, он может как создавать группы, так и быть приглашенным участником в группе. Сама по себе, группа не дает возможности обмена сообщениями. Для этого в группе создаются чаты, в которых и происходит обмен сообщениями между пользователями.

При создании группы создаются сущности, описывающие роль каждого пользователя в группе. Каждый пользователь будет обладать одной ролью в каждой группе. При создании чатов создаются сущности, определяющие права доступа для чатов, хранящие в себе списки пользователей, которые обладают или не обладают данным правом. В системе определены четыре права доступа для чатов:

* Возможность посещать чат
* Возможность писать сообщения в чате
* Возможность просматривать историю сообщений в чате
* Возможность отсылать файлы

Соответственно, для каждого чата создаются четыре сущности, описывающие права пользователей.

## Проектирование ПО и выбор архитектуры

В качестве основы построения программного обеспечения была выбрана клиент-серверная архитектура. Обмен данными между стороной клиента и стороной сервера реализуется при помощи асинхронных AJAX (Asynchronous JavaScript and XML — «асинхронный JavaScript и XML») запросов по сетевому протоколу HTTP. Так же для поддержания постоянного соединения между клиентом и сервером используется протокол WebSocket.

Архитектура серверного приложения состоит из следующих основных слоев:

* Уровень, содержащий себе маршруты для запросов от клиента на сервер, описывающий API (Application program interface) программы
* Уровень, который содержит в себе функции, которые выполняются при запросах
* Уровень, описывающий записи в базе данных
* Уровень, содержащий в себе промежуточные функции (middleware), которые проверяют права доступа пользователя в системе при запросах

Архитектура серверного приложения состоит из следующих основных слоев:

* Уровень отрисовки пользовательского интерфейса (UI – User Interface)
* Уровень, на котором реализуется основная логика и хранятся данные (State)
* Уровень, на котором исполняются запросы на сервер

### Алгоритм авторизации.

Для входа в систему, пользователь должен быть в ней зарегистрирован. Если он не зарегистрирован, то ему будет предложено пройти процесс регистрации. Схема алгоритма авторизации представлена на рисунке 3.1.

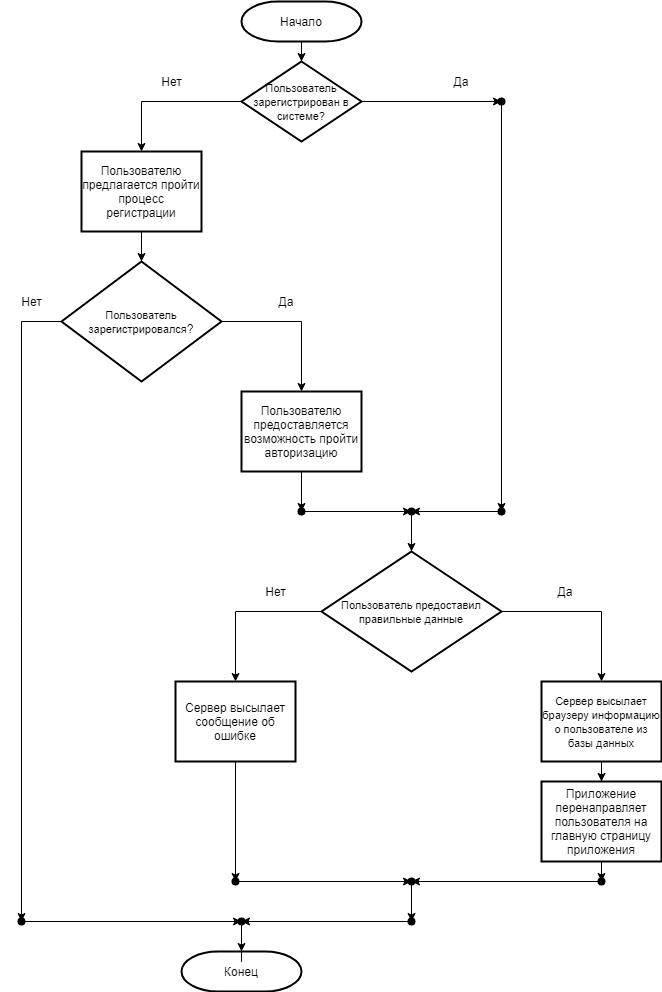


Рисунок 3.1 – Схема алгоритма авторизации пользователей

### Алгоритм создания групп

Далее, пользователю, для возможности обмена сообщениями, нужно участвовать в группе. Пользователь может быть приглашен в группу другого пользователя, либо создать группу самостоятельно. Схема алгоритма создания групп представлена на рисунке 3.2.



Рисунок 3.2 – Схема алгоритма создания групп

### Алгоритм создания чатов

Для возможности обмена сообщениями недостаточно просто создать группу, но также в ней должны присутствовать чаты, через которые будут реализовываться отправка и получение сообщений. Схема алгоритма создания чатов представлена на рисунке 3.3.

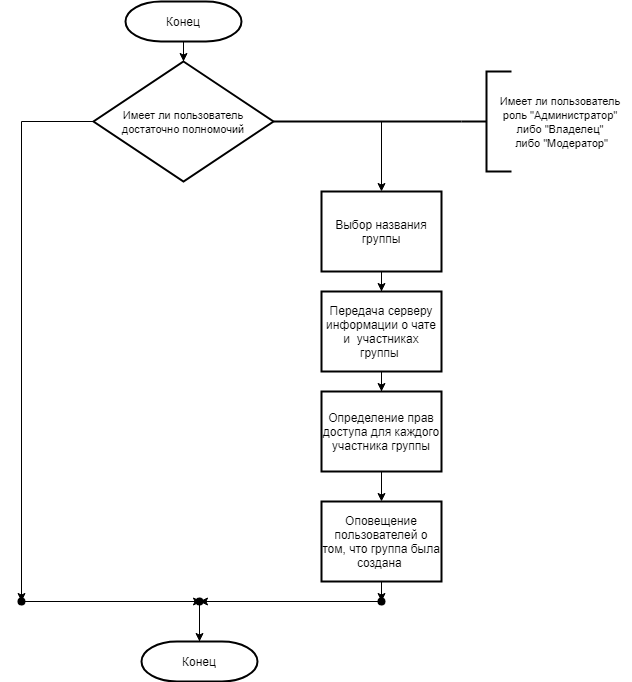


Рисунок 3.3 – Схема алгоритма создания чатов

### Алгоритм отправки сообщений

Пользователь, выбрав нужный чат, при условии, что он имеет нужные права доступа, может отправлять сообщения другим участникам группы. Схема алгоритма отправки сообщений представлена на рисунке 3.4.

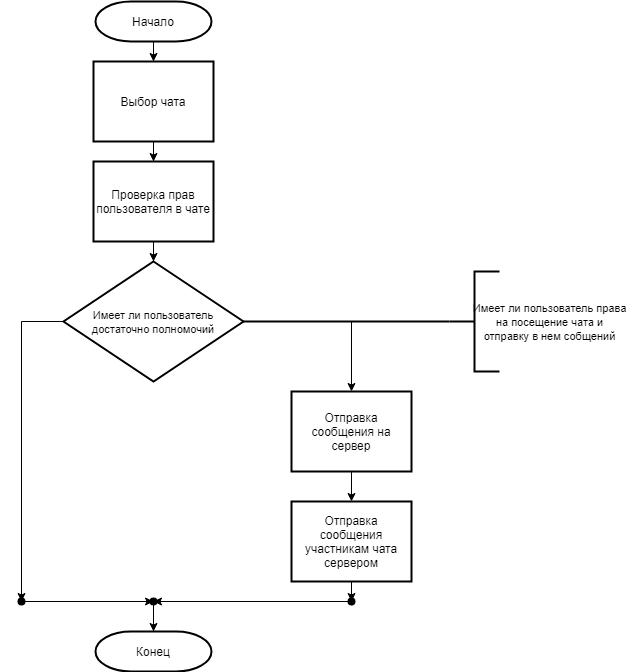


Рисунок 3.4 – Схема алгоритма отправки сообщений

## Программная реализация

Программа была реализована в виде веб-приложения, которое исполняется c помощью веб-браузер, используя следующие технологии:

* Программная платформа Node.js для реализации логики работы серверной части приложения. Используемый язык программирования – JavaScript.
* Библиотеку React.js для реализации клиентской части приложения. Используемый язык программирования – JavaScript.
* Нереляционная система управления базами данных – MongoDB.
* Библиотека Socket.io для использования протокола WebSocket обеспечивающего постоянное соединение между клиентской и серверной частями приложения

### Разработка структуры хранения данных

В рассматриваемом приложении используются следующие модели данных:

1) Users – модель данных, описывающая пользователей. Поля документов этой коллекции данных пердставлены в таблице 3.1.

Таблица 3.1 – Модель данных Users

|  |  |  |
| --- | --- | --- |
| Поле | Значение | Тип |
| Id | Идентифкатор пользователя | Строка |
| name | имя | Строка |
| email | адрес электронной почты | Строка |
| password | пароль | Строка |

2) Groups – модель данных, описывающая пользователей. Поля документов этой коллекции данных пердставлены в таблице 3.2.

Таблица 3.2 – Модель данных Groups

|  |  |  |
| --- | --- | --- |
| Поле | Значение | Тип |
| id | идентификатор группы | Строка |
| Partners | список, содержащий в себе идентификаторы участников группы | Массив |
| Name | название группы | Строка |
| Author | создатель группы | Строка |

3) User\_groupRoles – модель данных, описывающая пользователей. Поля документов этой коллекции данных представлены в таблице 3.3.

Таблица 3.3 – Модель данных groupRoles

|  |  |  |
| --- | --- | --- |
| Поле | Значение | Тип |
| id | идентификатор роли | Строка |
| Role | название роли | Строка |
| user\_name | идентификатор пользователя | Строка |
| group\_id | идентификатор группы | Строка |

4) Chanels – модель данных, описывающая пользователей. Поля документов этой коллекции данных пердставлены в таблице 3.4.

Таблица 3.4 – Модель данных Chanels

|  |  |  |
| --- | --- | --- |
| Поле | Значение | Тип |
| id | идентификатор роли | Строка |
| Name | название роли | Строка |
| author | идентификатор пользователя | Строка |
| group | идентификатор группы | Строка |
| canSee | Идентификатор документа, описывающего право, позволяющее просматривать историю сообщений канала | Строка |
| canSeeHistory | Идентификатор документа, описывающего право, позволяющее отправку файлов | Строка |
| canSendFile | Идентификатор документа, описывающего право, позволяющее отправку файлов | Строка |
| canWrite | Идентификатор документа, описывающего право, позволяющее посещать канал | Строка |

5) Rights – модель данных, описывающая пользователей. Поля документов этой коллекции данных представлены в таблице 3.5.

Таблица 3.5 – Модель данных Rights

|  |  |  |
| --- | --- | --- |
| Поле | Значение | Тип |
| id | идентификатор права | Строка |
| Prevelegion | значение, показывающее является ли данное право привелегированным | Логический |
| Whitelisted | значение, показывающее является ли список данного права белым списком, либо черным списком | Логический |
| List | список идентификаторов пользователей | Массив |
| Type | название права | Строка |
| group\_id | идентификатор группы | Строка |
| chanel\_id | идентификатор канала | Строка |

6) Messages– модель данных, описывающая пользователей. Поля документов этой коллекции данных представлены в таблице 3.6.

Таблица 3.6 – Модель данных Messages

|  |  |  |
| --- | --- | --- |
| Поле | Значение | Тип |
| id | идентификатор сообщения | Строка |
| Autor | идентификатор отправителя сообщения | Строка |
| Chanel\_id | идентификатор чата | Строка |

### Разработка серверной части

Серверная часть приложения использует принцип REST API (REST— архитектурный стиль взаимодействия компонентов распределённого приложения в сети) для передачи донных по протоколу HTTP и протокол WebSocket для обеспечения постоянного соединения между клиентом и сервером.

При использовании REST клиентская сторона отправляет сетевой запрос на сервер по протоколу HTTP по определенному маршруту. По маршруту определяется, какую логику будет выполнять сервер и каким будет ответ сервера на запрос. Для передачи данных в реальном времени по протоколу WebSocket либо сервер, либо клиент создают событие и отправляют данные на противоположную сторону, которая использует обработчик этого события.

#### Реализация системы регистрации и авторизации пользователей

При отправке клиентом POST запроса по маршруту http://localhost:8001/register, содержащий в своем теле объект, который содержит в себе имя, адрес электронной почты и пароль пользователя, на сервере вызывается функция, которая отвечает за регистрацию пользователей.

Сначала происходит проверка электронной почты: если пользователь с таким адресом уже зарегистрирован в системе, то сервер отправляет ответ со статусом 500, и сообщение о том, что пользователь с таким адресом уже зарегистрирован, если же пользователь с таким адресом не был ранее зарегистрирован, то пароль хешируется и создается новая запись в коллекции данных Users, тем самым сохраняя нового пользователя в системе.

При отправке клиентом POST запроса по маршруту http://localhost:8001/login, содержащий в своем теле объект, который содержит в себе адрес электронной почты и пароль пользователя, на сервере вызывается функция, которая отвечает за идентификацию пользователей в системе. В базе данных происходит поиск пользователя, который имеет переданный e-mail. Если такой пользователь не был найден, то сервер отправляет ответ со статусом 500, и сообщение о том, что пользователь с таким адресом не зарегистрирован. Иначе, если пользователь найден, то его пароль расшифровывается и сравнивается с введенным. Если пароли не совпадают друг с другом, то сервер отправляет ответ со статусом 500, и сообщение о том, что пользователь ввел неверный пароль. Если пароли совпали, то для пользователя формируется специальный ключ доступа – токен (для этого используется библиотека jsonwebtoken). Токен формируется посредством шифрования идентификатора пользователя, секретной фразы и числом, которое обозначает время жизни токена в миллисекундах. В результате шифрования токен представляет собой строку из 176 символов, состоящую из цифр, знаков пунктуации и букв латинского алфавита.

#### Создание системы записи, возврата, изменения и удаления групп

При отправке клиентом POST запроса по маршруту http://localhost:8001/groups на сервере запускается функция, которая создает объект, описывающий новую группу, в котором помимо, информации, полученной от браузера, содержится идентификатор группы, который записывается в поле group\_id каждому элементу массива ролей.

После этого создателю и другим участникам группы от сервера приходит оповещение о том, что новая группа была создана, и объект содержащий информацию об этой группе, и в списке групп на странице браузера появляется новая группа. Так же от сервера высылаются ключи, которые определяют права пользователя на использование чатов

При отправке клиентом GET запроса по маршруту http://localhost:8001/groups/Id, где Id – идентификатор пользователя, который произвел запрос, вызывается функция, которая ищет в базе данных все группы, в которых принимает участие пользователь с переданным Id. После чего сервер высылает ответ, в котором передает информацию о найденных группах.

#### Создание системы возврата, изменения и удаления групповых ролей

При отправке клиентом GET запроса по маршруту http://localhost:8001/roles\_all/groupId', где role - название роли, groupId – идентификатор группы, которую выбрал пользователь, вызывается функция, которая ищет в базе данных информацию о ролях каждого пользователя в выбранной группе. После чего сервер высылает ответ, в котором передает информацию о найденных пользователях.

При отправке клиентом PUT запроса по маршруту http://localhost:8001/ roles\_update/user\_name/groupId', содержащий в своем теле объект, который содержит в себе новую роль пользователя и имеющий параметры запроса: user\_name – имя выбранного пользователя, groupId – идентификатор группы, которую выбрал пользователь, вызывается функция, которая ищет в базе данных информацию о роле выбранного пользователя и изменяет ее на ту, которая была передана в теле запроса . После чего сервер высылает ответ, в котором содержится сообщение о том, что роль пользователя успешно изменена.

#### Создание системы записи, возврата, изменения и удаления чатов, а так же возврата правовых списков в чатах

При отправке клиентом POST запроса по маршруту http://localhost:8001/chanels, содержащий в своем теле объект, который содержит в себе название чата, идентификатор создателя чата, идентификатор группы и объекты, описывающие права доступа чата, на сервере вызывается функция, которая создает новый чат и права для этих чатов, путем создания новых записей в базе данных в коллекциях Chanels и Rights

При отправке клиентом GET запроса по маршруту http://localhost:8001/ chanels /userId/groupId, где userId – идентификатор пользователя, который произвел запрос, groupId – идентификатор группы, которую выбрал пользователь, вызывается функция, которая ищет в базе данных все чаты, которые принадлежат выбранной группе. Вместе с этим в базе данных находится информация о правах пользователей в каждом чате, по их идентификаторам, которые записаны как значения полей, определяющих связь между правом доступа чата и самим чатом. Это осуществляется при помощи метода populate библиотеки mongoose.

Листинг 3.1 – Функция возврата информации о чатах

router.get('/chanels/:user/:groupName/:id', async (req, res) => {

try {

let chanels = await Chanel

.find({ group: req.params.id })

.populate(['author', 'group', "canSee", 'canWrite', 'canSeeHistory', 'canSendFile']);

res.json({ chanels })

} catch (e) {

console.log(e)

}

})

После чего сервер высылает ответ, в котором передает информацию о найденных чатах.

#### Создание функции изменения правовых списков в чатах

Для изменения прав доступа участников чата, клиент может отправить PUT запрос по маршруту http://localhost:8001/ right/update/:rightId ', содержащий в своем теле объект, который обязательно содержит в себе свойство list - массив идентификаторов пользователей и может содержать новое значение для свойства prevelegion, определяющие, что право будет включать в свой список не всех участников группы, а только тех, которые переданы в свойстве list, и свойства whitelisted, определяющее, что список будет «Белым» либо «Черным». Запрос имеет параметр rightId – идентификатор изменяемого права. При получении такого запроса, сервер вызывает функцию, которая находит нужное право в коллекции Rights и, если свойство prevelegion было передано, то в записи будет изменено свойство prevelegion, whitelisted и list на те, которые были переданы в теле запроса, иначе, будет изменено только свойство list

#### Создание функции, проверяющей права доступа пользователя, пославшего запрос на сервер

Для того, чтобы пользователь, не имеющий определенных прав, не мог отсылать запросы на сервер, нужно проверять каждый запрос на то, есть ли у пользователя доступ к ресурсу, который он запрашивает. Для этого нужна специальная функция, которая, в качестве своих параметров принимает информацию о запросе. В заголовках запроса должен быть передан Токен, который выдается пользователю после авторизации. Если в заголовке нет токена, то сервер вернет браузеру сообщение об ошибке. Если же токен был передан, то функция его расшифрует, используя метод verify из библиотеки jsonwebtoken, в результате чего должен быть получен идентификатор пользователя, выполнившего запрос. В первую очередь, данная функция, используя идентификатор пользователя должна проверить является ли пользователь авторизованным, далее, в зависимости от запроса, функция может проверить роль пользователя в группе или его права в чате.

#### Создание системы записи, возврата, изменения и удаления сообщений

Функция, предназначенная для записи сообщения, получает в теле запроса идентификатор отправителя сообщения, идентификатор чата, по которому было отправлено сообщение, и текст сообщения. На основе этих данных создается запись в базе данных в коллекции Messages и, созданный объект передается браузеру в качестве ответа сервера. Далее, это сообщение снова отсылается на сервер по протоколу WebSocket, и сервер, по тому же протоколу, рассылает сообщение всем пользователям, которые имеют право доступа на посещение чата, по которому было отправлено сообщение

### Разработка клиентской части

Клиентская часть веб-приложения выполнена при помощи JavaScript библиотеке – React, которая позволяет представлять пользовательский интерфейс как набор компонент, которые работают так же, как и обычные JavaScript функции, которые принимают в качестве своих аргументов могут принимать объект с любыми свойствами и которые возвращают React-элементы, описывающие, что мы хотим увидеть на экране. Это позволяет использовать одни и те же элементы интерфейса в различных местах программы несколько раз. При изменении входных параметров, компонент вызывается заново, уже с новыми параметрами, тем самым провоцируя процесс перерисовки компоненты – на экране пользователя появится тот же элемент интерфейса, но уже предоставляющий другую информацию.

Листинг 3.2 – Пример React-компонента

function Welcome(props) {

return <h1>Привет, {props.name}</h1>;

}

React использует специальный синтаксис для написания интерфейсов называемый JSX (JavaScript XML), с помощью которого можно совмещать и разметку, и логику при описании интерфейса.

#### Создание основных интерфейсов программы

Для начала, необходимо разработать страницы веб-приложения и основные элементы для взаимодействия с программой с помощью JSX и языка стилей CSS, предназначенного для стилизации элементов веб-страниц. Первое что нужно сделать пользователю для использования ПО – это зарегистрироваться, соответственно, первая страница, которую нужно создать – это страница регистрации пользователей, а далее, по ее подобию страницы регистрации, нужно создать страницу авторизации. Страницы регистрации и авторизации представлены на рисунках 3.5 и 3.6 соответственно.

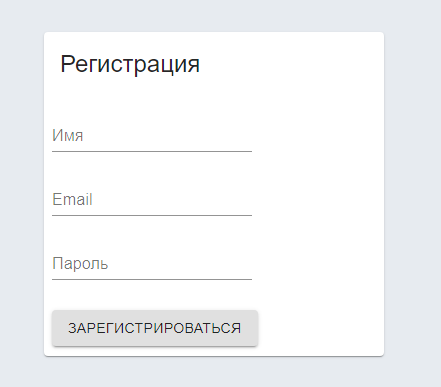


Рисунок 3.5 – Форма регистрации

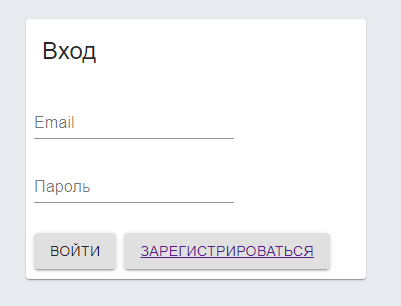


Рисунок 3.6 – Форма авторизации

После авторизации, сайт предоставляет страницу, на которой показан список групп, в которых участвует пользователь. Список групп продемонстрирован на рисунке 3.7.

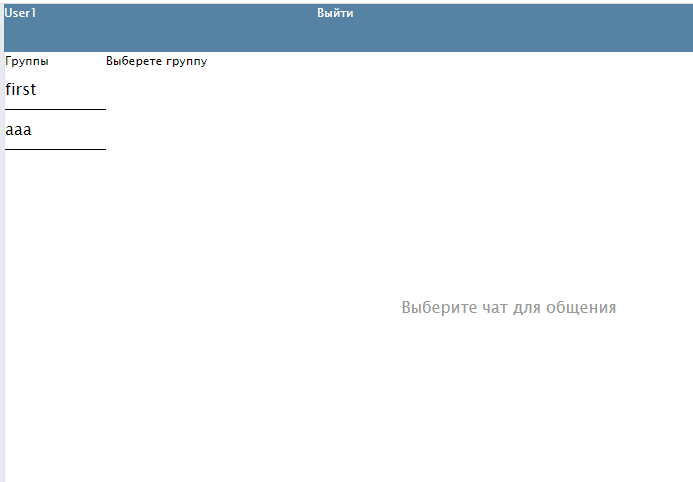


Рисунок 3.7 – Список групп

После выбора группы пользователь сможет увидеть список чатов группы. Список чатов продемонстрирован на рисунке 3.8.

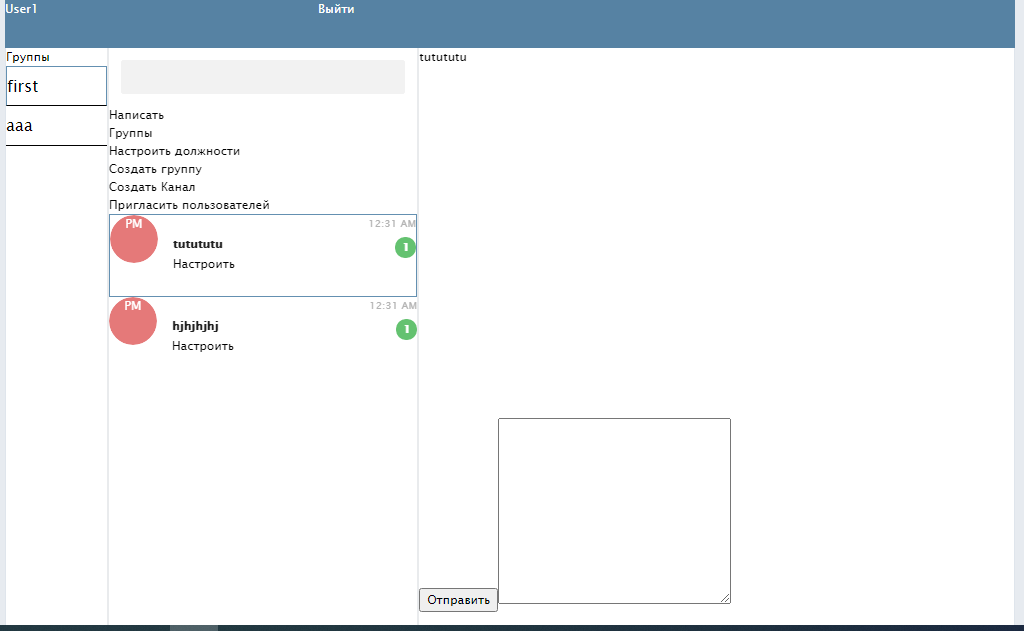


Рисунок 3.8 – Интерфейс списка чатов и формы отправки сообщений

#### Разработка логики отображения страниц

При каждом попадании на сайт либо при каждой перезагрузке страницы сайта, приложение проверяет авторизован ли пользователь или нет. Перед отрисовкой страницы, запускается функция, в которой проверяется, есть ли в lockalStotage свойство token, значением которого является токен, который сервер высылает браузеру после успешно пройденной авторизации. Если этого свойства нет, то пользователю будет доступно посещать только страницы авторизации и регистрации, иначе, пользователю будет доступна страница просмотра групп и чатов

#### Разработка логики процессов регистрации и авторизации

Чтобы пользователь мог пользоваться информационной системой, он должен быть в ней зарегистрирован. Для регистрации пользователю нужно ввести имя, которым он будет обозначен в системе, ввести адрес электронной почты, и придумать пароль к новой учетной записи. Для увеличения надежности пароля, пользователь не сможет сохранить пароль, который содержит менее 6 символов. После ввода данных, эти данные отправляются на сервер.

Авторизация

Для того, чтобы войти в систему пользователю нужно ввести свой e-mail и пароль. Сначала эти идентификаторы проходят процесс валидации: проверяется, корректен ли e-mail, и содержит ли пароль минимальное количество символов. Если валидация прошла успешно, то идентификаторы отправляются на сервер, где происходит процесс идентификации.

При успешной идентификации клиенту приходит от сервера ответ, в котором содержится информация о пользователе и токен, который сохраняется в объекте localStorage – хранилище данных браузера.

#### Разработка логики создания, получения и отображения групп

После успешной авторизации приложение перенаправляет пользователя на страницу, на которой он может увидеть список групп, в которых он состоит (<http://localhost:3000/chat>). Так же пользователю предоставляется возможность создать собственную группу. Для этого, в первую очередь, пользователь должен нажать на кнопку «Создать группу», после чего он увидит форму создания группы с элементами:

- Поле для ввода названия группы;

- Список пользователей, которых можно пригласить в группу;

- Кнопка отправки формы.

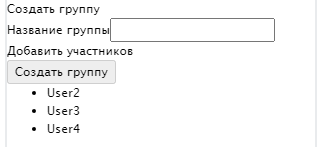


Рисунок 3.9 – форма создания группы.

Так же, после нажатия на кнопку создается массив roles, первым элементом которого становится объект - Role, который определяет роль пользователя в новой группе, со свойствами user\_id, значением которого становится идентификатор создателя группы, и role, значением которого становится строка «owner», что означает, что данный пользователь будет обладать ролью «Владелец» в группе. При нажатии на пользователя из списка, идентификатор этого пользователя записывается в массив partners и, помимо этого, создается объект Role, у которого в поле user\_id записан идентификатор пользователя, а в поле role – строка «partner», что означает, что данный пользователь будет обладать ролью «Участник» в группе.

После нажатия на кнопку отправки формы на сервер посылается AJAX запрос, имеющий метод POST, и передающий массив пользователей группы и массив ролей этих пользователей

От сервера приходит оповещение создателю и другим участникам группы о том, что новая группа была создана, и объект содержащий информацию об этой группе, и в списке групп на странице браузера появляется новая группа. Так же от сервера высылаются ключи, которые определяют права пользователя на использование чатов.

#### Проверка прав пользователя в группах

При нажатии на кнопку «Настроить должности» приложение перенаправит пользователя на страницу настройки ролей

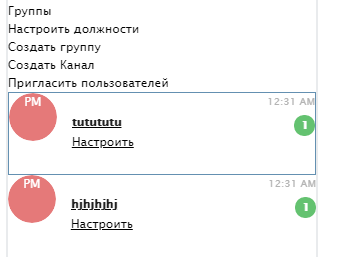


Рисунок 3.10 – Интерфейс группы.

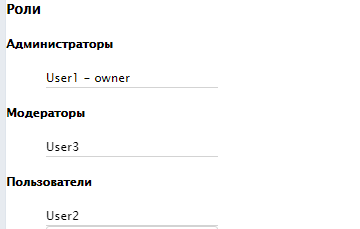


Рисунок 3.11 – Список пользователей и их ролей в группе.

При нажатии на пользователя в списке пользователей появляется контекстное меню с возможными вариантами назначения ролей и кнопка удаления выбранного пользователя из группы

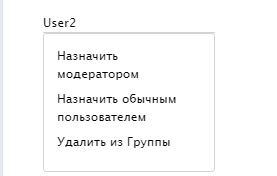


Рисунок 3.12 – контекстное меню.

При нажатии на кнопку назначения должности отправляется запрос на сервер, где находиться документ в коллекции User\_grouproles в котором изменяется значение поля role.

#### Разработка логики создания, получения и отображения чатов

Кликнув на группу, можно увидеть список чатов группы и кнопку для создания нового чата, после на экране появится форма создания чата с элементами:

- Поле для ввода названия чата;

- Кнопка настройки прав;

- Кнопка отправки формы.

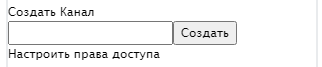


Рисунок 3.13 – форма создания чата.

После ввода названия чата и нажатия кнопки «Создать» на сервер посылается AJAX запрос, имеющий метод POST и передающий объект, для создания новой записи в базе данных в коллекции Canels

Листинг 3.2 – Запрос на создание чата

let chanel = await axios.post('http://localhost:8001/chanels', {

            name: text,

            author: props.author,

            group: props.selectedGroup.\_id,

            canSee: new Right('canSee',[...namesInGroups]),

            canWrite: new Right('canWrite',[...namesInGroups]),

            canSeeHistory: new Right('canSeeHistory',[...namesInGroups]),

            canSendFile: new Right('canSendFile',[...namesInGroups]),

       })

#### Проверка прав пользователя в чате

При нажатии на кнопку «Настроить», появляется форма настроек прав доступа

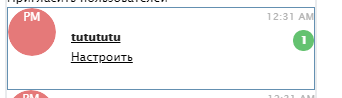


Рисунок 3.14 – чат.



Рисунок 3.15 – интерфейс настройки прав доступа.

При нажатии на кнопку, которая предлагает изменить список пользователей, появляется окно со списком пользователей

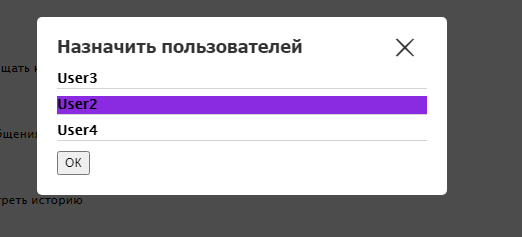


Рисунок 3.16 – форма создания списка пользователей.

Идентификаторы выбранных пользователей записываются в массив, и после нажатия на кнопку «ОК» этот массив отправляется на сервер, где определяет список пользователей обладающих, или лишенных выбранным правом

#### Разработка логики отправки сообщений

После выбора чата, пользователю предоставляется возможность отправки сообщения. После того, как пользователь ввел сообщение и нажал кнопку отправить, на сервер отправляется объект, содержащий текст сообщения, идентификатор пользователя-отправителя и идентификатор чата. Ответом, сервер высылает объект сообщения, сохраненного в базе данных и имеющего идентификатор. Далее, этот объект отсылается всем пользователям, у которых открыто приложение и у которых есть доступ на чтение сообщений из чата. Эти пользователи получают объект сообщения и, если чат, через который было послано сообщение, открыт у них в браузере, то программа записывает объект в конец массива, в котором содержатся сообщения чата, после чего происходит перерисовка компонента, отвечающего за отображение чата, в результате чего пользователи смогут увидеть ново полученное сообщение в чате.