СОДЕРЖАНИЕ

[Введение 8](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783152)

[1.1. Описание предметной области и постановка задачи 9](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783153)

[1.2. Технологии реализации 10](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783154)

[1.3. Описание практической реализации 11](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783155)

[1.3.1. Создание моделей 11](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783156)

[1.3.2. Создание репозиториев 15](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783157)

[1.3.3. Создание сервисов 17](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783158)

[1.3.4. Создание контроллеров 24](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783159)

[1.3.5. Создание представлений 31](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783160)

[Заключение 33](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783161)

[Список использованных источников 34](file:///C:\Users\walla\Downloads\Telegram%20Desktop\212017_b1-IFST21%20(2).docx#_Toc140783162)

# Введение

Создание сайта для киберклуба становится все более актуальной задачей с ростом популярности компьютерных игр. Такой сайт позволит оптимизировать процесс регистрации игроков, осуществлять запись на тренировки, публиковать последние новости, создавать команды и записываться на турниры.

Целью проекта является создание удобного интерфейса для пользователей и участников киберклуба.

Задачи:

1. Проектирование базы данных для хранения информации о пользователях, командах, турнирах, новостях;
2. Написание программной логики для регистрации и авторизации пользователей, записи на тренировки и турниры, создания команд и публикации новостей на сайте.
3. Разработка интерфейса сайта, который будет интуитивно понятен для пользователей и удобен в использовании.

## Описание предметной области и постановка задачи

Разрабатываемое приложение - корпоративное веб-решение с системой личных кабинетов для киберклуба. Такое приложение позволяет решать проблемы, связанные с оптимизацией процессов регистрации игроков, записью на тренировки и турниры, а также улучшить взаимодействие между участниками клуба и организаторами мероприятий.

Предметная область - киберспорт и деятельность киберклубов, которые являются популярными местом сбора среди людей, увлекающихся компьютерными играми. Решение, которое предлагается, помогает оптимизировать процесс работы киберклубов и улучшить взаимодействие пользователей на основе правильного подбора функций, таких как автоматизация процесса записи на соревнования и т. д. Это позволяет участникам клуба быстрее и проще находить необходимую информацию и управлять своими аккаунтами.

Актуальность и перспективы развития в предметной области оцениваются положительно, так как киберспорт продолжает расти в популярности и количество киберклубов также увеличивается. В связи с этим, разработка такого приложения является востребованной задачей.

Аналоги существующих решений могут быть изучены для определения функциональности, которой может не хватать в рассматриваемом корпоративном веб-решении. Например, уже существующие приложения для киберклубов могут не предоставлять возможности для создания личных кабинетов участников или не иметь возможности для автоматизации процессов записи на соревнования и тренировки.

Необходимые функции приложения включают:

1. Система регистрации и авторизации для участников киберклуба.
2. Создание личных кабинетов для участников и возможность управления ими.
3. Возможность записи на тренировки и соревнования.
4. Публикация новостей о киберспорте и мероприятиях, проводимых киберклубом.
5. Создание и управление командами, участвующими в соревнованиях.

Таким образом, разработка корпоративного веб-решения с системой личных кабинетов может значительно упростить работу участников киберклуба и организаторов мероприятий, улучшить коммуникацию между ними и повысить участие в киберспортивных мероприятиях.

## Технологии реализации

Для разработки монолитного приложения для киберклуба были выбраны следующие технологии: язык программирования Java, база данных PostgreSQL и шаблонизатор Thymeleaf.

В качестве основной технологии для разработки приложения был выбран язык Java. Для работы с базами данных была использована библиотека Hibernate. Для создания контроллеров и представлений был использован фреймворк Spring MVC.

Выбор использованных технологий при разработке приложения был обоснован следующими критериями:

**Язык программирования Java** — это широко используемый язык программирования, который обладает большой популярностью благодаря своей простоте, масштабируемости и универсальности. Java позволяет разрабатывать высокопроизводительные и масштабируемые приложения, а также запускать их на различных операционных системах и устройствах.

**База данных PostgresSQL** — это открытая реляционная база данных, которая обеспечивает высокую производительность, надежность и безопасность. Она может хранить огромное количество данных и легко работать с ними с помощью SQL-запросов.

**Thymeleaf** — это шаблонизатор для веб-приложений, который позволяет создавать отзывчивые и интерактивные веб-страницы. Thymeleaf обеспечивает простой, удобный и эффективный способ создания веб-страниц, что упрощает работу разработчиков.

## Описание практической реализации

Было разработано монолитное приложение на Java для киберклуба, которое включает следующие модели: User, Team, News, Training, Tournament, Participant. Для каждой из моделей были созданы соответствующие репозитории для работы с базами данных, сервисы, контроллеры и представления.

Структура приложения:

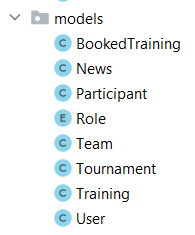
1. Модели: User, Team, News, Training, Tournament, Participant.
2. Репозитории для работы с базами данных для каждой модели.
3. Сервисы для бизнес-логики приложения.
4. Контроллеры для обработки HTTP-запросов и передачи управления на соответствующие сервисы.
5. Представления для отображения данных для пользователей.

Был использован подход "Model-View-Controller" для организации кода в проекте. Модели представляют данные приложения, репозитории служат для получения и сохранения этих данных в базе данных. Сервисы содержат бизнес-логику приложения, а контроллеры обрабатывают HTTP-запросы и передают управление на соответствующие сервисы.

## Создание моделей

Модели создаются для представления данных в приложении. Они позволяют описать структуру данных, определить свойства и атрибуты, а также способы взаимодействия с данными. Модели используются как в целях отображения данных в пользовательском интерфейсе, так и в бизнес-логике приложения. Они также служат для упрощения работы с базами данных и позволяют сократить объем кода, необходимого для работы с данными.

В данном проект были созданы следующие модели:



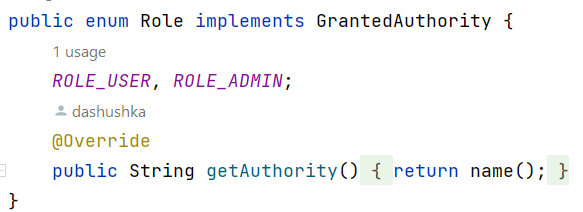
*Рис 1. Модели*

Модель User – пользователь. Она предназначена как для регистрации и авторизации, так для и функциональности.



*Рис 2. Модель User*

Модель Role – роль пользователя (User) на сайте. В данном проекте есть две роли – обычный пользователь и администратор. Администратору доступно больше действия, таких как: добавление новостей, турниров, тренировок, создание команд.



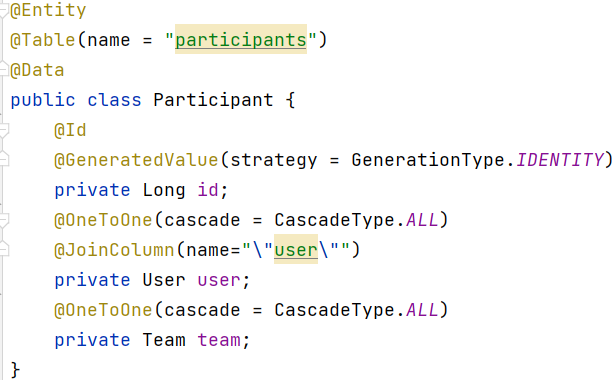
*Рис 3. Модель Role*

Модель Team – команда. Ее создает администратор, и только потом добавляет в нее участников.



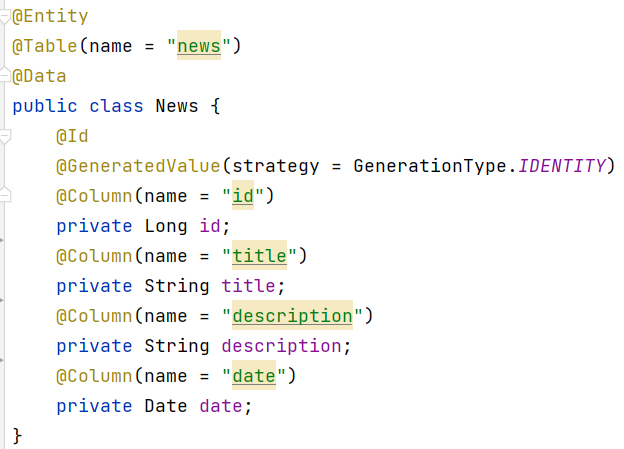
*Рис 4. Модель Team*

Модель Participant – участник. Данная модель создана для связи между пользователем и командой. С помощью этой модели имеется возможность реализовать добавление участника в ту или иную команду.

**

*Рис 5. Модель Participant*

Модель News – новости. Новости можно добавлять администратору.



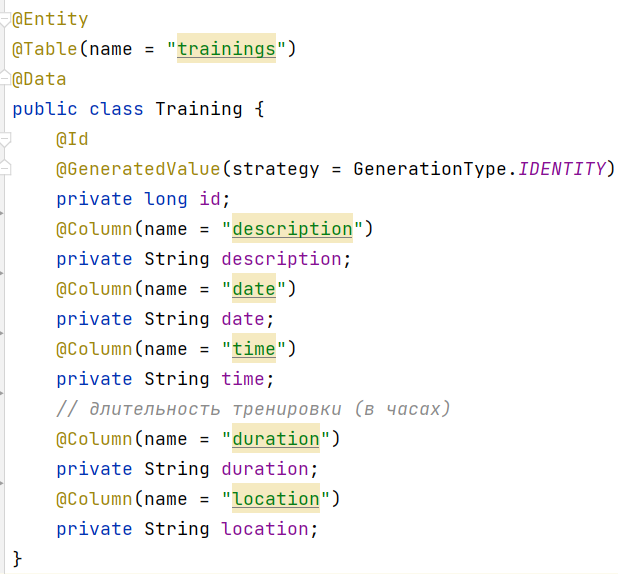
*Рис 6. Модель News*

Модель Tournament – турнир. На него можно зарегистрироваться команде. Новый турнир может создать администратор.



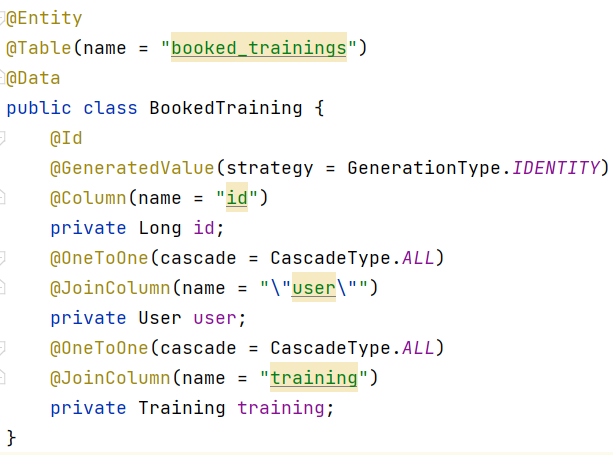
*Рис 7. Модель Tournament*

Модель Training – тренировка. На тренировку может записаться пользователь.



*Рис 8. Модель Training*

Модель BookedTraining – забронированная тренировка. Данная модель создана для связи между Training и User. Когда пользователь записывается на тренировку, в базе данных сохраняется связь между моделями.

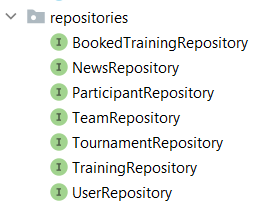


*Рис 9. Модель BookedTraining*

## Создание репозиториев

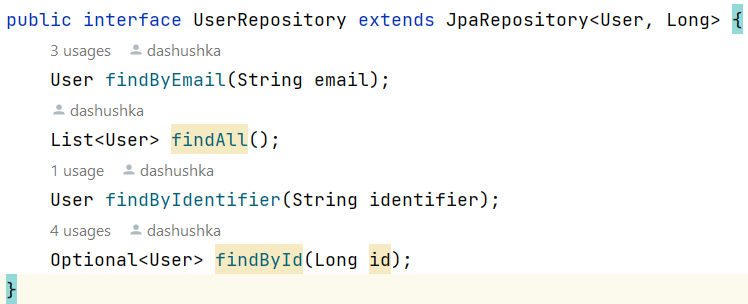
Репозитории создаются для организации взаимодействия между приложением и базой данных. Они предоставляют абстрактный интерфейс для доступа к данным, скрывая детали реализации и обеспечивая более удобный способ работы с базой данных. Репозитории содержат методы для добавления, удаления, изменения и получения данных из базы данных. Приложение может использовать репозитории для выполнения запросов к базе данных и получения результатов запросов в виде моделей данных. Создание репозиториев также позволяет улучшить архитектуру приложения, разделив функциональность базы данных и бизнес-логики.

В данном проекте были созданы следующие репозитории:



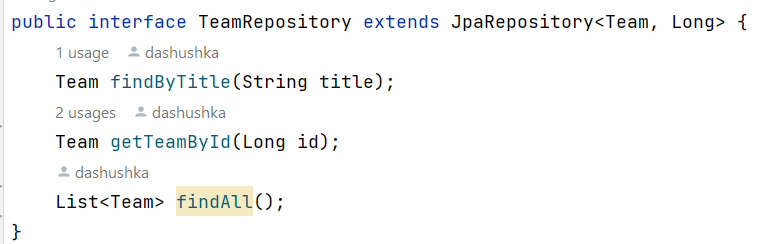
*Рис 10. Репозитории*

UserRepository – репозиторий, связанный с User. В нем находятся такие методы, как: поиск пользователя по email, по identifier, по ID, и поиск абсолютно всех пользователей.

**

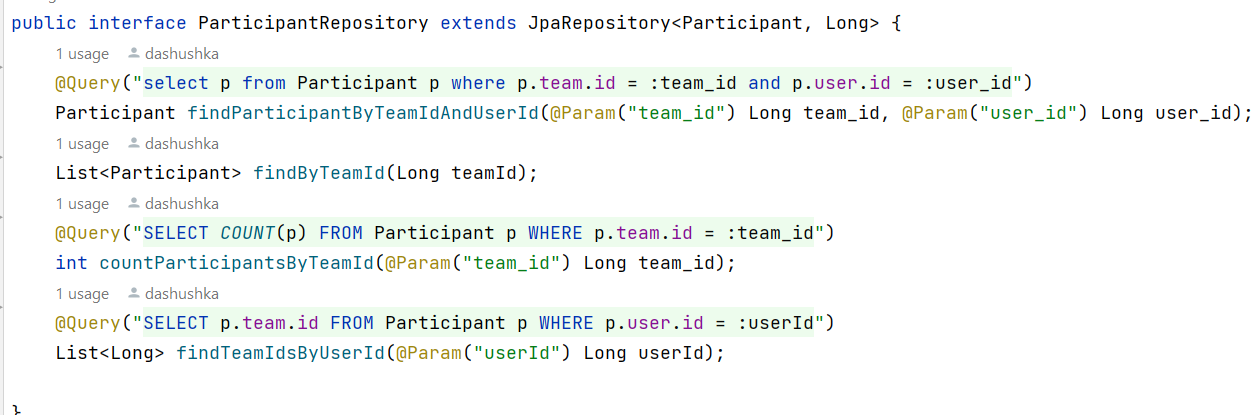
*Рис 11. UserRepository*

TeamRepository – репозиторий, связанный с моделью Team. У него есть следующие методы: найти команду по названию, найти команду по ID и найти все команды.

**

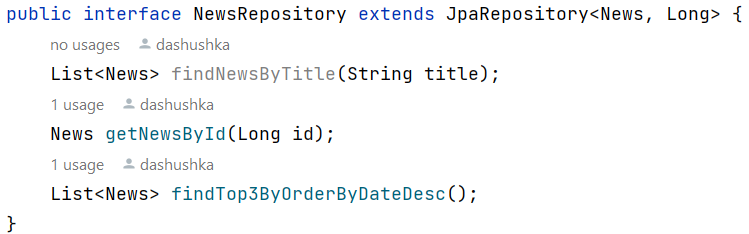
*Рис 12. TeamRepository*

ParticipantRepository – репозиторий, связанный с моделью Participant. У него есть методы, для которых пришлось самостоятельно прописывать запросы к базе данных.

**

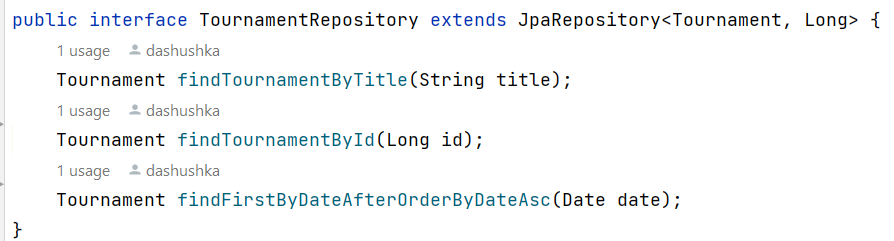
*Рис 13. ParticipantRepository*

NewsRepository – репозиторий, относящийся к модели News. У него есть такие методы, как: найти новость по названию, по ID, а также метод для нахождения последних трех по дате новостей.

**

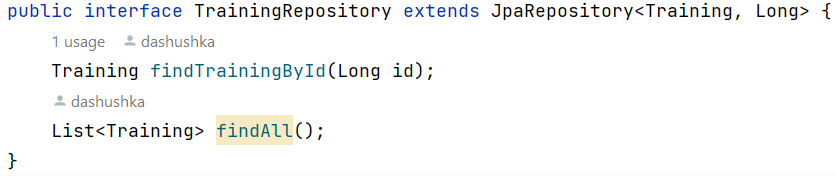
*Рис 14. NewsRepository*

TournamentRepository – репозиторий, связанный с моделью Tournament. У него есть следующие методы: найти турнир по названию, по ID, а также метод по нахождению ближайшего по дате турнира.

**

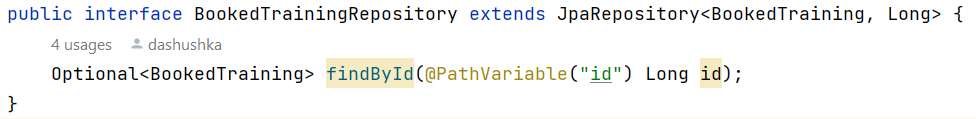
*Рис 15. TournamentRepository*

TrainingRepository – репозиторий, относящийся к модели Training. У него есть такие методы, как: найти тренировку по ID и найти все тренировки.

**

*Рис 16. TrainingRepository*

BookedTrainingRepository – репозиторий, связанный с моделью BookedTraining. У него есть метод по нахождению забронированной тренировки по ID.

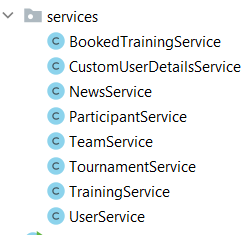
**

*Рис 17. BookedTrainingRepository*

## Создание сервисов

Сервисы создаются для реализации бизнес-логики приложения. Они используются для обработки запросов, полученных от контроллеров приложения. В сервисах происходит обработка данных, логика взаимодействия с базой данных, обработка ошибок и другие операции, связанные с бизнес-логикой приложения. Создание сервисов позволяет сделать код приложения более модульным и улучшить его архитектуру, разделив отображение данных от бизнес-логики, а также обеспечивает повторное использование кода для различных контроллеров.

В данном проекте были созданы следующие сервисы:

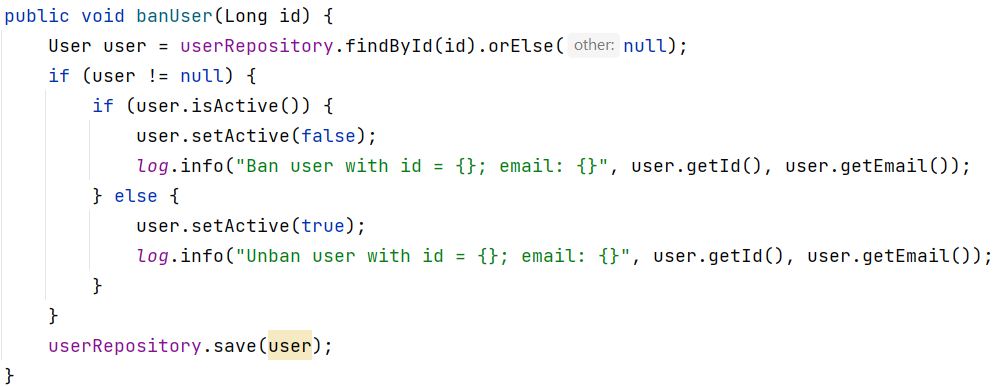


*Рис 18. Сервисы*

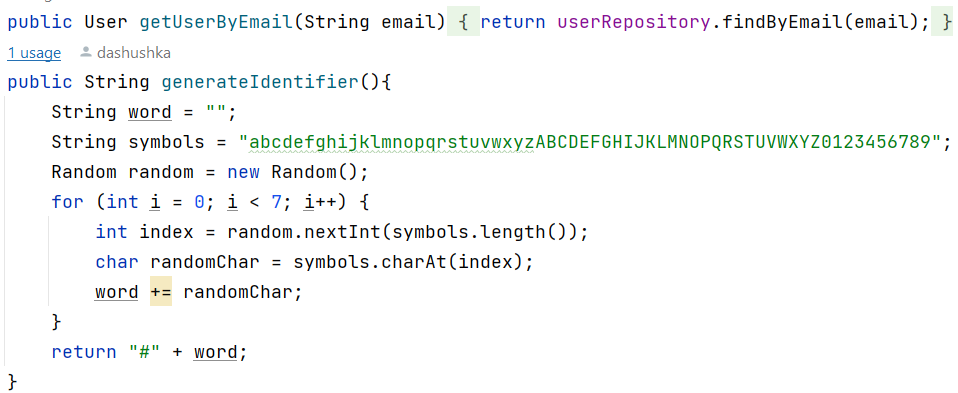
UserService представляет собой сервисный класс, ответственный за взаимодействие с моделью User в приложении. Он реализует методы для получения, удаления и добавления пользователей в базу данных, используя соответствующий репозиторий для работы с базой данных. UserService может использоваться в контроллерах приложения для обработки запросов от пользователей, например, для создания нового пользователя или удаления пользователя.

**

*Рис 19. UserService*

**

*Рис 20. UserService. Продолжение*

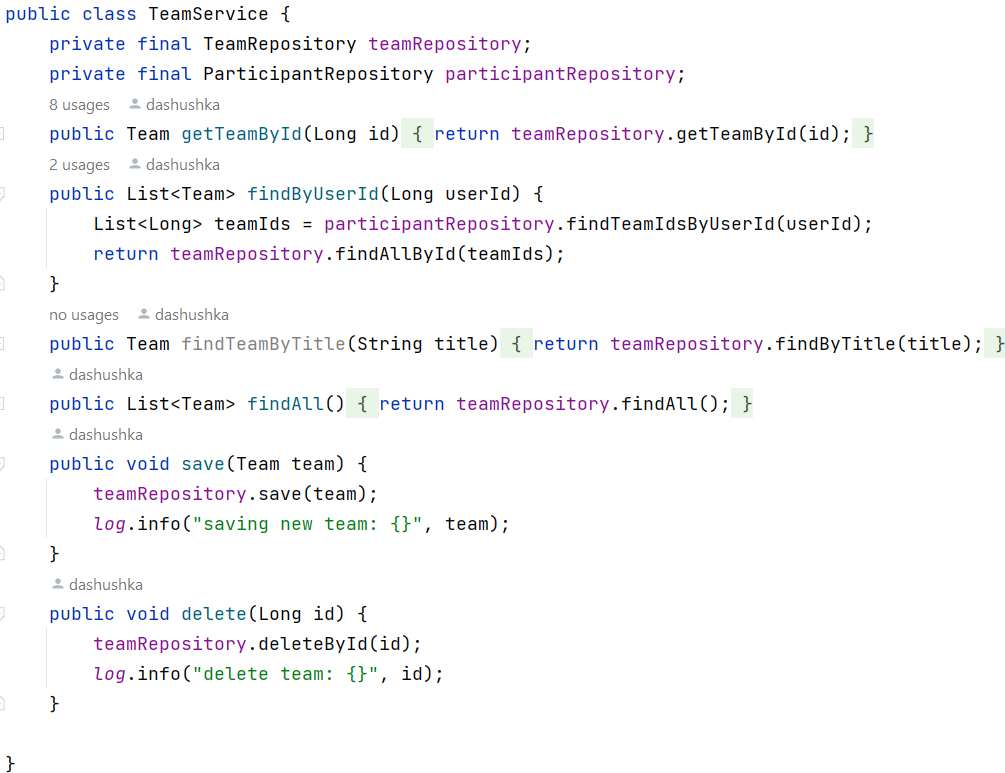
**

*Рис 21. UserService. Продолжение*

TeamService предназначен для реализации бизнес-логики, связанной с моделью Team в приложении.

В сервисе реализованы методы, которые обращаются к репозиторию (TeamRepository и ParticipantRepository) для выполнения различных операций над данными. Например, метод getTeamById(Long id) возвращает объект команды по его идентификатору, а метод findByUserId(Long userId) возвращает список команд, в которых указанный пользователь является участником.

Метод findTeamByTitle(String title) находит команду по названию, а методы findAll() возвращают список всех команд в базе данных. Метод save(Team team) сохраняет команду в базе данных, а метод delete(Long id) удаляет команду с заданным идентификатором.



*Рис 22. TeamService*

Сервисный класс ParticipantService отвечает за реализацию бизнес-логики, связанной с моделью Participant в приложении.

В сервисном классе ParticipantService определен метод addParticipantToTeam, который добавляет участника (объект типа Participant) в команду. В начале метода производится поиск участника, используя метод findParticipantByTeamIdAndUserId. Если такой участник не найден, то происходит проверка количества участников в команде (quantity) и, если количество еще не достигнуто, участник добавляется в команду с помощью вызова метода save из ParticipantRepository.

Если участник уже существует в команде, то выбрасывается исключение IllegalArgumentException.

Также в сервисе определен метод getParticipantsByTeamId, который возвращает список участников по идентификатору команды.

Сервисный класс ParticipantService использует несколько репозиториев (TeamRepository, UserRepository, ParticipantRepository), чтобы получать и изменять данные в базе данных. Он может использоваться в контроллерах как часть архитектуры приложения для обработки запросов и ответов от клиента, например, для добавления участников в команду или получения списка участников команды для отображения на странице участников.

**

*Рис 23. ParticipantService*

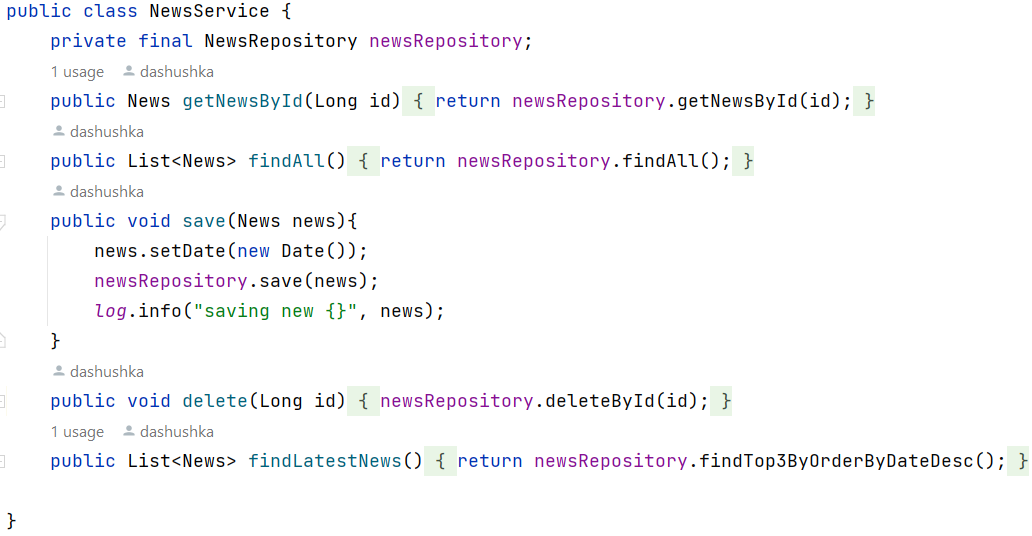
Сервисный класс NewsService отвечает за реализацию бизнес-логики, связанной с моделью News в приложении.

В сервисном классе NewsService определен метод getNewsById(Long id), который возвращает объект новости по его идентификатору. Кроме того, метод findAll() возвращает список всех новостей в базе данных.

Метод save(News news) принимает новость и сохраняет ее в базе данных. Для новости устанавливается текущая дата, а затем она сохраняется с помощью метода save из NewsRepository.

Метод delete(Long id) удаляет новость с заданным идентификатором из базы данных с помощью метода deleteById из NewsRepository.

Метод findLatestNews() возвращает список последних трех новостей, отсортированных по дате.

**

*Рис 24. NewsService*

Сервисный класс TournamentService отвечает за реализацию бизнес-логики, связанной с моделью Tournament в приложении.

В сервисном классе TournamentService определены методы для работы с сущностью Турнира в базе данных. Метод findTournamentByTitle(String title) возвращает турнир по его названию, метод findTournamentById(Long id) возвращает турнир по его идентификатору, метод save(Tournament tournament) сохраняет турнир в базе данных, метод findAll() возвращает список всех турниров, метод delete(Long id) удаляет турнир с заданным идентификатором из базы данных.

Метод findNearestTournament() возвращает самый ближайший турнир. Для этого происходит предварительное получение текущей даты и времени с помощью объекта класса Date. Затем в репозитории TournamentRepository вызывается метод findFirstByDateAfterOrderByDateAsc, который возвращает первый турнир, дата которого идет после текущей даты и времени, отсортированный по возрастанию даты.

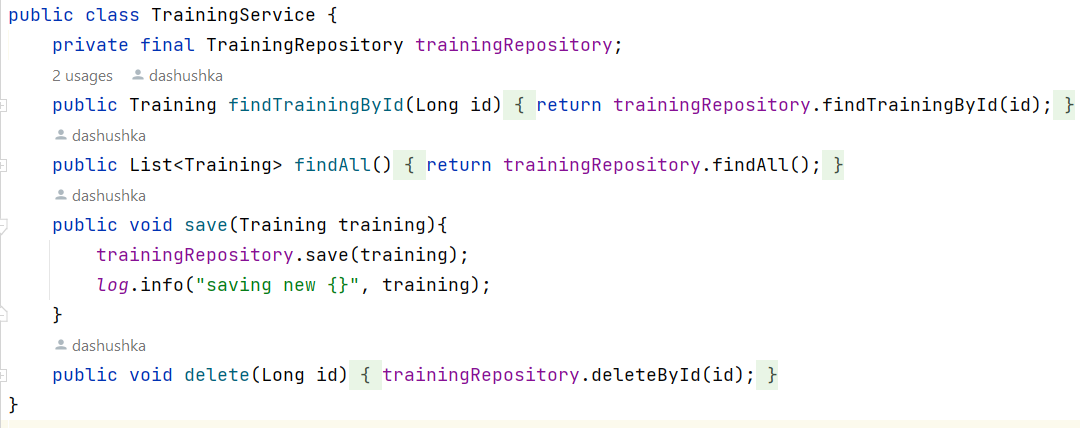
**

*Рис 25. TournamentService*

Сервисный класс TrainingService отвечает за реализацию бизнес-логики, связанной с моделью Training в приложении.

В сервисном классе TrainingService определены методы для работы с сущностью Тренировки в базе данных. Метод findTrainingById(Long id) возвращает тренировку по её идентификатору, метод findAll() возвращает список всех тренировок в базе данных. Метод save(Training training) сохраняет тренировку в базе данных, а метод delete(Long id) удаляет тренировку с указанным идентификатором из базы данных.

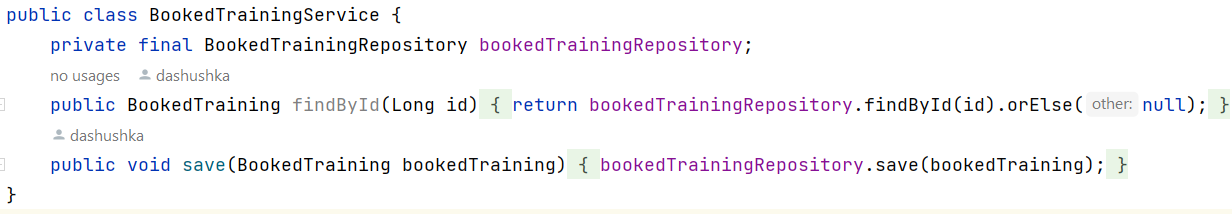
Сервисный класс TrainingService использует репозиторий TrainingRepository, чтобы получать и изменять данные в базе данных.

**

*Рис 26. TrainingService*

Сервисный класс BookedTrainingService отвечает за реализацию бизнес-логики, связанной с моделью BookedTraining в приложении.

В сервисном классе BookedTrainingService определены методы для работы с сущностью Забронированной тренировки в базе данных. Метод findById(Long id) возвращает забронированную тренировку по её идентификатору, метод save(BookedTraining bookedTraining) сохраняет заказанную тренировку в базе данных.

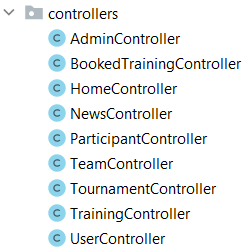
**

*Рис 27. BookedTrainingService*

## Создание контроллеров

Контроллеры создаются для обработки запросов от клиента и отправки ответов в соответствии с логикой приложения. Они помогают разделить код приложения на уровни и упрощают процесс взаимодействия с пользователем.

В данном проекте были созданы следующие контроллеры:



*Рис 28. Контроллеры*

Контроллер UserController отвечает за обработку запросов, приходящих от клиента, и направление полученных запросов на обработку соответствующим методам внутри контроллера.

Контроллер UserController содержит ряд методов, связанных с обработкой запросов, таких как методы login(), registration() и profile().

Методы login() и registration() отвечают за обработку запросов для страницы входа и регистрации соответственно. Метод createUser(User user) - сохраняет пользователя в базе данных после его регистрации.

Метод profile() отвечает за получение данных о пользователе из базы данных и передачи этой информации в представление, что позволяет отобразить информацию в личном кабинете пользователя.

Кроме того, класс UserController также использует аннотации @GetMapping, @PostMapping, @Secured и @PreAuthorize для предоставления доступа только зарегистрированным пользователям и аутентифицированным пользователям, а также для обеспечения безопасности приложения. безопасности приложения.



*Рис 29. UserController*



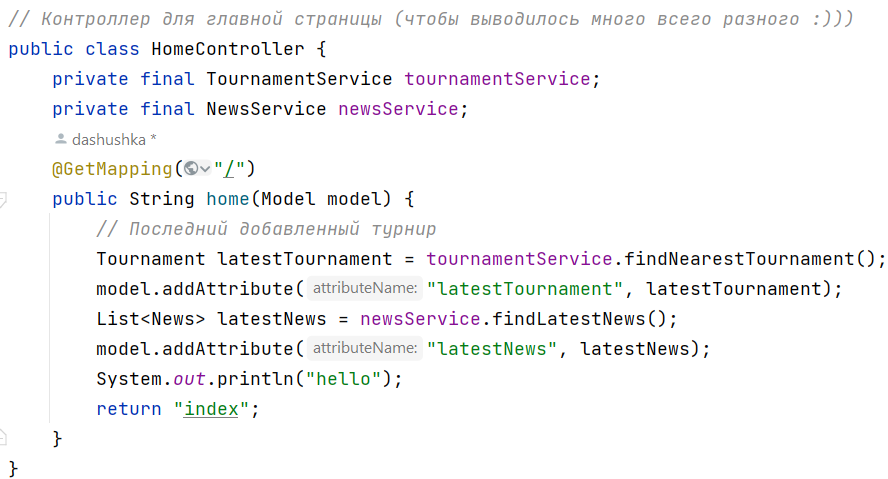
*Рис 30. UserController. Продолжение*

Данный класс "HomeController" является контроллером, который обрабатывает запросы для главной страницы ("/").

Метод "home" помечен аннотацией "@GetMapping", которая указывает на то, что этот метод обрабатывает GET запросы, посылаемые на адрес "/".

Внутри метода "home" выполняются следующие операции:

1. Получение объекта сервиса "TournamentService" и "NewsService", которые будут использованы для получения данных о последнем добавленном турнире и последних новостях.
2. Получение последнего турнира из БД и добавление его в атрибуты модели "latestTournament".
3. Получение последних новостей из БД и добавление их в атрибуты модели "latestNews".
4. Возврат имени представления ("index"), которое будет отображено в ответ на запрос.



*Рис 31. HomeController*

AdminController является контроллером, который обрабатывает запросы, связанные с административной частью сайта.

Аннотация "@PreAuthorize" на классе указывает на то, что только пользователи с ролью "ROLE\_ADMIN" могут получить доступ к этому контроллеру.

Метод "admin" обрабатывает GET запросы, посылаемые на адрес "/admin". Он получает список всех пользователей с помощью сервиса "UserService" и добавляет его в атрибуты модели "users". Затем он возвращает имя представления ("admin/index"), которое будет отображено пользователю.

Метод "userBan" обрабатывает POST запросы, посылаемые на адрес "/admin/user/ban/{id}". Он принимает id пользователя, которого нужно забанить, и вызывает метод "banUser" у объекта сервиса "UserService". Затем он выполняет перенаправление на страницу администратора.

Метод "userInfo" обрабатывает GET запросы, посылаемые на адрес "/user/{user}". Он принимает объект "User", соответствующий запрошенному пользователю, и добавляет его в атрибуты модели "user". Затем он возвращает имя представления ("admin/info-about-users"), которое будет отображено пользователю.

Таким образом, контроллер "AdminController" позволяет администраторам получить список пользователей, забанить пользователей и просмотреть информацию о конкретном пользователе. Доступ к этому контроллеру имеют только пользователи с ролью "ROLE\_ADMIN".



*Рис 32. AdminController*

NewsController является контроллером, который обрабатывает запросы, связанные с новостями сайта.

Метод "newsInfo" обрабатывает GET запросы, посылаемые на адрес "/news/{id}", где id - это идентификатор новости. Он получает объект новости из БД с помощью сервиса "NewsService" и добавляет его в атрибуты модели "news". Затем он возвращает имя представления ("news/info"), которое будет отображено пользователю.

Метод "listNews" обрабатывает GET запросы, посылаемые на адрес "/news/list". Если в запросе указан параметр "title", то метод получит список новостей с этим заголовком, иначе получит список всех новостей с помощью сервиса "NewsService". Список новостей добавляется в атрибуты модели "news". Затем он возвращает имя представления ("news/index"), которое будет отображено пользователю.

Метод "createNews" обрабатывает GET запросы, посылаемые на адрес "/news/create". Он отображает форму для создания новости, путем возврата имени представления ("news/create").

Метод "createNews" обрабатывает POST запросы, посылаемые на адрес "/news/create". Он получает объект новости из параметров запроса и вызывает метод "save" у объекта сервиса "NewsService" для сохранения новости в БД. Затем он выполняет перенаправление на главную страницу.

Метод "deleteNews" обрабатывает POST запросы, посылаемые на адрес "/news/delete/{id}", где id - это идентификатор новости для удаления. Он вызывает метод "delete" у объекта сервиса "NewsService" и выполняет перенаправление на главную страницу.



*Рис 33. NewsController*

TeamController является контроллером, который обрабатывает запросы, связанные с командами.

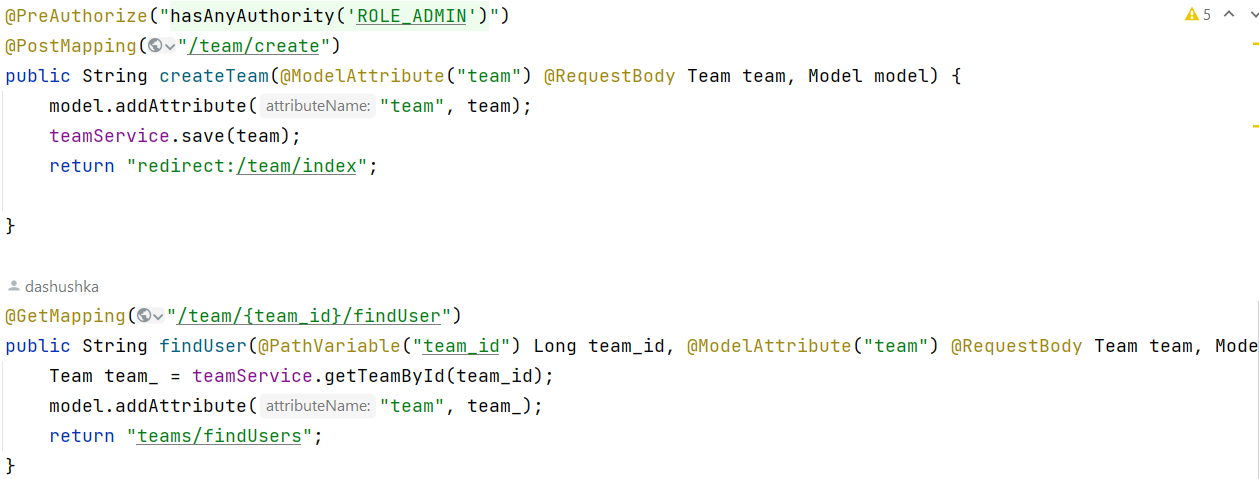
Контроллер обеспечивает функциональность для создания, поиска и удаления команд, а также для добавления и удаления участников в команды.

Некоторые из методов контроллера доступны только для пользователей с ролью "ROLE\_ADMIN", в то время как другие доступны для любых пользователей с ролью "ROLE\_USER" или "ROLE\_ADMIN".

Контроллер также обеспечивает возможность просмотра информации о конкретной команде и ее участниках, используя методы "teamInfo" и "getTeamInfo".



*Рис 34. TeamController*



*Рис 35. TeamController. Продолжение*



*Рис 36. TeamController. Продолжение*

TournamentController является контроллером, который обрабатывает запросы, связанные с турнирами.

Контроллер обеспечивает функциональность для создания, просмотра, удаления турниров и подписки на них.

Некоторые из методов контроллера доступны только для администраторов, в то время как другие доступны для любых пользователей.

Контроллер использует "TournamentService" для взаимодействия с данными о турнирах.



*Рис 37. TournamentController*

TrainingController" является контроллером, который обрабатывает запросы, связанные с тренировками.

Контроллер обеспечивает функциональность для создания, просмотра и удаления тренировок.

Контроллер использует "TrainingService" для взаимодействия с данными о тренировках.



*Рис 38. TrainingController*

BookedTrainingController является контроллером, который обрабатывает запросы, связанные с забронированными тренировками.

Контроллер обеспечивает функциональность для бронирования тренировки и добавления информации о бронировании в базу данных.

Авторизация пользователя происходит в методах "bookedTraining" и "addBookedTraining" с помощью аннотации "@PreAuthorize".



*Рис 39. BookedTrainingController*

## Создание представлений

Представления .html с использованием Thymeleaf создаются для рендеринга динамических веб-страниц. Thymeleaf является шаблонизатором, который позволяет создавать шаблоны веб-страниц, заменяя в них фрагменты на динамические данные, которые могут быть получены из контроллеров веб-приложений, используя синтаксис выражений Thymeleaf. Представления, созданные с использованием Thymeleaf, могут быть многократно использованы в различных страницах приложения для отображения данных, что упрощает разработку и поддержку веб-приложений.

Например, в данном проекте на главную страницу выводятся данные из базы данных с помощью Thymeleaf: ближайший турнир и три последние новости.



*Рис 40. Пример главной страницы приложения*

## Заключение

В рамках данного проекта было разработано монолитное приложение на языке программирования Java с использованием базы данных PostgreSQL для киберклуба. Были успешно реализованы следующие функции: регистрация и авторизация пользователей, опубликование и удаление новостей, создание команд и добавление в них зарегистрированных участников, создание турниров и возможность записи на них, создание тренировок и возможность записи на них.

Для обеспечения безопасности пользовательских данных в приложении были использованы системы аутентификации и авторизации. Было также реализовано управление объединениями пользователей в команды и возможностью участия в турнирах. Организовано создание тренировок и возможность записи на них, позволяя пользователям развивать свои навыки в играх.

В качестве базы данных была использована PostgreSQL, что позволило управлять информацией о пользователях, новостях, командах, турнирах, тренировках и других объектах приложения, обеспечивая быстрый и эффективный доступ к данным для отображения на сайте.

В процессе разработки были применены передовые технологии и инструменты, что позволило обеспечить производительность, масштабируемость и безопасность приложения. Наличие всех реализованных функций обеспечивает удобный и практичный интерфейс пользователям, а также создает удобные условия для организации многопользовательских мероприятий.

## Список использованных источников

1. Как сделать проект на Java Spring Boot? [Электронный ресурс] // Хабр URL: https://habr.com/ru/articles/586176/ (дата обращения: 25.06.23).
2. Serving Web Content with Spring MVC [Электронный ресурс] // Spring URL: https://spring.io/guides/gs/serving-web-content/ (дата обращения: 25.06.23).
3. Spring Security [Электронный ресурс] // Baeldung URL: https://www.baeldung.com/spring-deprecated-websecurityconfigureradapter (дата обращения: 27.06.23).
4. Web [Электронный ресурс] // Spring URL: https://docs.spring.io/spring-boot/docs/current/reference/html/web.html (дата обращения: 30.06.23).
5. Spring MVC Tutorial [Электронный ресурс] // Baeldung URL: https://www.baeldung.com/spring-mvc-tutorial (дата обращения: 02.07.23).
6. Hibernate и Spring Boot [Электронный ресурс] // Devmark URL: https://devmark.ru/article/spring-boot-hibernate (дата обращения: 05.07.23).
7. Spring Boot CRUD Application with Thymeleaf [Электронный ресурс] // Baeldung URL: https://www.baeldung.com/spring-boot-crud-thymeleaf (дата обращения: 10.07.23).
8. Spring Security Architecture [Электронный ресурс] // Spring URL: https://spring.io/guides/topicals/spring-security-architecture/ (дата обращения: 13.07.23).
9. Spring MVC + Thymeleaf 3.0: New Features [Электронный ресурс] // Baeldung URL: https://www.baeldung.com/spring-thymeleaf-3 (дата обращения: 15.07.23).
10. Spring Web Annotations [Электронный ресурс] // Baeldung URL: https://www.baeldung.com/spring-mvc-annotations (дата обращения: 17.07.23).