# ВВЕДЕНИЕ

В современной разработке программного обеспечения неотъемлемую часть занимают системы управления проектами. Ни один проект среднего масштаба невозможен без системы контроля и учета времени, такие системы служат основным инструментарием в управлении командой, постановке и планировании времени. Они помогают в планировании архитектуры, во время разработки легко можно узнать состояние проекта взглянув на количество выполненных задач. Так же программисту в любое время доступны описания всех задач на всех этапах разработки, что упрощает представление целостной картины проекта. Они предоставляют полную интеграцию с системами контроля версий, что позволяет лидеру разработки легко видеть все изменения в ветках разработчиков, а так же производить их слияние в нужный момент. Все это позволяет с легкостью вести разработки по гибким метологиям.

В данном курсовом проекте я исследую подход к исчислению рабочего времени программиста, рассмотрю с практической точки зрения удобство разных подходов их преимущества и недостатки. Так же разработаю систему на основе собственного метода учета индивидуального времени работы, рассмотрю его удобство использования. Реализованный метод должен быть универсальным решением к любой из багтрекинговых систем, с возможностью расширяться для любой из ихних API. В основу проекта ляжет система «YouTrack» так как она имеет на данный момент большую популярность среди отечественных разработчиков, имеет очень хороший и понятный интерфейс, а так же продвигается по лицензии GNU GPL для не более 10 пользователей, что приемлемо в рамках курсового проекта.

Цель данного курсового проекта разработать web-приложение для подсчета рабочего времени программиста.

# ОСНОВНАЯ ЧАСТЬ

* 1. Анализ предметной области

Систем контроля и учета времени большое количество, и все они имеют свои достоинства и недостатки. В большинстве из них очень много лишнего и избыточного функционала, который только отвлекает от быстрого и удобного использования в повседневной жизни. Такие системы обычно используют подход логирования или тайм-трекинга времени.

Как показывает практика, эти подходы имеют свои недостатки. Первый, связанный с логированием, чаще всего программисты завышают свои оценки или вовсе забывают указать отработанное время, после чего восстанавливая его из памяти, оценивают его с большими погрешностями. Второй недостаток связан с тайм-трекингом, чаще всего программист работает над несколькими задачами одновременно или использует парное программирование, включив таймер на одной задаче, он может переключиться на другую, что снова приводит к большим погрешностям

Все эти недостатки ведут в будущем к неправильному планированию, и ошибках в сроках разработки программных продуктов, что в свою очередь ведет к большим финансовым потерям. Так же бизнес аналитик при планировании должен видеть реальную картину временных рамок, на которую нужно ориентироваться при разработке бизнес модели проекта. Работа же менеджеров заключается в том, что бы знать точные временные затраты у каждого отдельно взятого программиста, это приводит к минимизации рисков на проектах. При правильном учете времени работника мы можем знать примерное время, которое ему нужно на реализацию той или иной задачи.

Тайм трекинговые системы обычно заточены под гибкие методологии разработки такие как agile или scrum. Они нуждаются в точном хронометраже времени, тогда подход работает точно и дает максимальный результат.

В некоторых компаниях практикуют оплату труда почасово, что приводит к необходимости дисциплинировать себя во время замера рабочего времени. Но и это не всегда помогает точно отработать восемь часов в день, что приводит к демотивации персонала, потому что в конце месяца он не набирает полное отработанное время, при использовании метода тайм-трекинга, а при использовании методов логирования обычно завышает его.

Так же эти системы используются при ведении тестирования проекта как баг-трекинговые, при этом так же важно знать точно потраченное время на исправление той или иной ошибки.

## 1.2 Постановка задачи

Передо мной была поставлена цель – разработка системы контроля и учета времени в виде web-приложения с возможностью интеграции YouTrack аккаунта в систему.

В системе необходимо реализовать:

* возможность регистрировать нового пользователя.
* вход в систему.
* настройки пользователя, интеграция с системой Youtrack.
* просмотр проектов из системы YouTrack.
* просмотр задач и информации о них.
* возможность просмотра личной статистики.
* возможность трекинга времени, и обратная интеграция.

Для разработки такой программы необходимо решить следующие функциональные задания:

а) обзор и анализ существующих веб-приложений и сайтов

б) разработка алгоритмической структуры программы

в) разработка программного обеспечения программы

## 1.3 Анализ существующих веб-приложений и сайтов

### 1.3.1 Assembla

Данный сервис использует систему логирования времени, хорошо документирован, имеет интеграцию с системами контроля версий. К недостаткам можно отнести дороговизну, а так же то что система не устанавливается на сервер компании, не имеет API или интеграций с другими сервисами. Одним из существенных недостатков является отсутствие русского перевода, так же плохо понятный интерфейс.

### 1.3.2 Trello

Сервис имеет не плохую фильтрацию по задачам и поиск, так же относительно недорогой, использует для замеров времени логирование. На этом его достоинства заканчиваются. Из недостатков это отсутствие переводов на другие языки, плохой не понятный интерфейс, нет возможности интеграции с другими сервисами, отсутствие API.

### 1.3.3 Битрикс24

Данный сервис предоставляет компания «Битрикс 1С» имеет ряд больших преимуществ. Это хорошо понятный интерфейс, так же он распространяется бесплатно, имеет хорошее REST API и неплохую документацию полностью на русском языке. Из недостатков это нет интеграции с системами контроля версий, нет возможности развертывать на собственном сервере, частые сбои в работе серверов, и постоянные обновления с изменением функционала. Так же сервис не имеет доски задач что затрудняет восприятия целостной картины проекта.

### 1.3.4 YouTrack

Сервис YouTrack имеет ряд достоинств, по которым он был выбран для реализации данного курсового проекта. Хорошо переведенный интерфейс на множество языков, полная интеграция с системами контроля версий, удобный и понятный интерфейс, а так же наличие удобной доски задач. Весь сайт построен на технологии AJAX что очень удобно для менеджера, в реальном времени следить за состоянием проекта. Имеет хорошо документированное REST API, возможность двусторонней интеграции с помощью GET/POST запросов. Так же сервис распространяется по лицензии GNU GPL для не более 10 пользователей, есть возможность развертывания системы на стороннем сервере, все эти преимущества и легли в основу выбора для данного курсового проекта.

## 1.4 Техническое задание на разработку программного продукта

### 1.4.1 Основания для разработки

Разработка данного проекта ведётся на основании задания курсового проектирования «Организация баз данных и знаний».

Данная информационная система реализуется в рамках курсового проекта на тему «Разработка системы учета и контроля рабочего времени ИТ-компании».

Руководитель проекта Смидович Л.С. доцентом кафедры «Программирование информационных систем» высшего учебного заведения НАУ «ХАИ».

### 1.4.2 Предназначение разработки

Данная система предназначена для повседневного использования в разработке программного обеспечения, трекинга времени. Это осуществляется с помощью тайм-трекера написанного на javascript, а так же системы хранения данных MySQL. Данный сервис интегрируется с YouTrack посредствам REST API. И предстает для конечного пользователя конечно системой.

### 1.4.3 Порядок контроля и приёма

Контроль осуществляется конечными пользователями системы, подключёнными на этапе тестирования системы.

Приём комплекса осуществляется после его полной установки и настройки под конкретных пользователей и короткого курса по обучению пользователей.

После окончания разработки системы должны быть проведены следующие виды испытаний: тестирование на защиту от неправильного ввода; тестирование на полноту обмена информацией между различными приложениями.

## 1.5 Описание языка (среды) программирования

PHP — [скриптовый язык](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством [хостинг-провайдеров](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3) и является одним из лидеров среди языков, применяющихся для создания [динамических веб-сайтов](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B0%D0%B9%D1%82). В области веб-программирования, в частности серверной части, PHP — один из популярных [сценарных языков](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) (наряду с [JSP](https://ru.wikipedia.org/wiki/JSP), [Perl](https://ru.wikipedia.org/wiki/Perl) и языками, используемыми в [ASP.NET](https://ru.wikipedia.org/wiki/ASP.NET)), благодаря своей простоте, скорости выполнения, богатому функционалу, [кроссплатформенности](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) и распространению исходных кодов на основе [лицензии PHP](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_PHP).

В настоящее время PHP используется сотнями тысяч разработчиков. Согласно рейтингу корпорации TIOBE, базирующемся на данных поисковых систем, в сентябре 2015 года PHP находился на 6 месте среди языков программирования. К крупнейшим сайтам, использующим PHP, относятся [Facebook](https://ru.wikipedia.org/wiki/Facebook), [Wikipedia](https://ru.wikipedia.org/wiki/Wikipedia) и др. Входит в [LAMP](https://ru.wikipedia.org/wiki/LAMP) — распространённый набор программного обеспечения для создания и [хостинга](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3) [веб-сайтов](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82) ([Linux](https://ru.wikipedia.org/wiki/Linux" \o "Linux), [Apache](https://ru.wikipedia.org/wiki/Apache), [MySQL](https://ru.wikipedia.org/wiki/MySQL), PHP).

HTML — стандартный [язык разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) документов во [Всемирной паутине](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Большинство [веб-страниц](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) содержат описание разметки на языке HTML (или [XHTML](https://ru.wikipedia.org/wiki/XHTML)). Язык HTML интерпретируется [браузерами](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80); полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Язык HTML является приложением [SGML](https://ru.wikipedia.org/wiki/SGML) (стандартного обобщённого языка разметки) и соответствует международному стандарту [ISO](https://ru.wikipedia.org/wiki/ISO) 8879. Во всемирной паутине HTML страницы, как правило, передаются браузерам от сервера по протоколам [HTTP](https://ru.wikipedia.org/wiki/HTTP) или [HTTPS](https://ru.wikipedia.org/wiki/HTTPS), в виде простого текста или с использованием [шифрования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Laravel — бесплатный веб-[фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA" \o "Фреймворк) с [открытым кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), предназначенный для разработки с использованием архитектурной модели [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller). Ключевые особенности, лежащие в основе архитектуры Laravel:

* *Пакеты* ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *packages*) — позволяют создавать и подключать модули в формате [Composer](https://ru.wikipedia.org/wiki/Composer" \o "Composer) к приложению на Laravel. Многие дополнительные возможности уже доступны в виде таких модулей.
* *Eloquent*[*ORM*](https://ru.wikipedia.org/wiki/ORM) — реализация [шаблона проектирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [ActiveRecord](https://ru.wikipedia.org/wiki/ActiveRecord" \o "ActiveRecord) на PHP. Позволяет строго определить отношения между объектами [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Стандартный для Laravel построитель запросов Fluent поддерживается ядром Eloquent.
* *Логика приложения* — часть разрабатываемого приложения, объявленная либо при помощи контроллеров, либо маршрутов ([функций-замыканий](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))). Синтаксис объявлений похож на синтаксис, используемый в [каркасе](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D0%BA%D0%B0%D1%80%D0%BA%D0%B0%D1%81) [Sinatra](https://ru.wikipedia.org/wiki/Sinatra" \o "Sinatra).
* *Обратная маршрутизация* связывает между собой генерируемые приложением ссылки и маршруты, позволяя изменять последние с автоматическим обновлением связанных ссылок. При создании ссылок с помощью именованных маршрутов Laravel автоматически генерирует конечные [URL](https://ru.wikipedia.org/wiki/URL).
* [*REST*](https://ru.wikipedia.org/wiki/REST)*-контроллеры* — дополнительный слой для разделения логики обработки GET- и POST-запросов [HTTP](https://ru.wikipedia.org/wiki/HTTP).
* *Автозагрузка классов* — механизм автоматической загрузки классов [PHP](https://ru.wikipedia.org/wiki/PHP) без необходимости подключать файлы их определений в *include*. Загрузка по требованию предотвращает загрузку ненужных компонентов; загружаются только те из них, которые действительно используются.
* *Составители представлений* ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *view composers*) — блоки кода, которые выполняются при генерации представления (шаблона).
* [*Инверсия управления*](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Inversion of Control*) — позволяет получать экземпляры объектов по принципу обратного управления. Также может использоваться для создания и получения [объектов-одиночек](https://ru.wikipedia.org/wiki/Singleton) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *singleton*).
* *Миграции* — [система управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8) для [баз данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Позволяет связывать изменения в коде приложения с изменениями, которые требуется внести в структуру БД, что упрощает развёртывание и обновление приложения.
* [*Модульное тестирование*](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) (*юнит-тесты*) — играет очень большую роль в Laravel, который сам по себе содержит большое число тестов для предотвращения [регрессий](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) (ошибок вследствие обновления кода или исправления других ошибок).
* *Страничный вывод* ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *pagination*) — упрощает генерацию страниц, заменяя различные способы решения этой задачи единым механизмом, встроенным в Laravel.

# 2 СПЕЦИАЛЬНАЯ ЧАСТЬ

## 2.1 Общие требования для программного продукта

Система написанная на фреймворке «Laravel» она может быть развернута на любом сервере с использованием данных требований:

Общие ограничения Back-end:

* Сервер Apache/2.4;
* Версия клиента базы данных: mysqlnd 5.5.0;
* PHP 5.6
* Composer 2.3.0

Общие ограничения Front-end:

* Веб-браузер поддерживающий JavaScript.

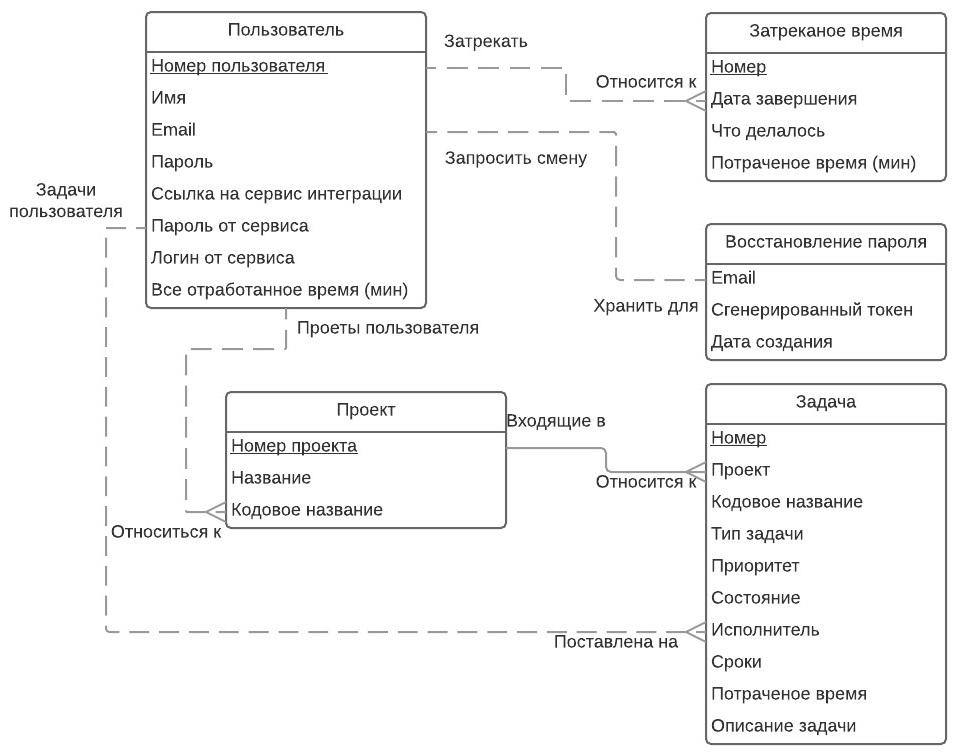
## 2.2 Описание общего алгоритма хронометража

Одной из целей курсового проекта было создание нового подхода к измерению рабочего времени. В основу легло объединение двух самых распространённых алгоритмов хронометража. Первый это логирование времени при котором работник сам указывает отработанные часы для каждой задачи, что ведет к многим неточностям. Второй подход тайм-трекинг, при котором работник запускает таймер на каждой отдельной задаче, он имеет больше преимуществ в точности, но не исключает, что программист может переключиться на более приоритетную задачу и таким образом вызовет большие неточности в потраченном времени. Созданный подход должен минимизировать погрешности, он заключается в том что работник накапливает время в так называемый банк времени, при этом он может указывать все время что он делает что бы не забыть потом распределить по задачам. После накопленных часов он может распределить нужное ему количество времени по задачам, это дает большую свободу для переключения с задачи на задачу и минимизирует погрешности.

## 2.3 ER – диаграмма моделей сущностей

В данном курсовом проекте реализуется модель сущностей показано на рисунке 2.1

Рисунок 2.1 – Диаграмма моделей сущностей



Из рисунка мы видим главную модель «Пользователь» она реализует экземпляр сущности каждого отдельно взятого пользователя в системе его основные данные, которые указываются при регистрации, а так же после первого входа в систему. Модель «Восстановление пароля» эта сущность создается при запросе восстановления пароля для каждого отдельно взятого пользователя. Экземпляр сущности «Затреканое время» относиться к определенному пользователю и несет в себя часть замеренного времени пользователем, связь с сущностью «Пользователь» один к многим. Так же есть модель «Проект» она хранит в себе данные о проектах полученные из API, относиться к пользователю как один пользователь ко многим проектам. Модель «Задача» имеет две связи, первая это с моделью «Пользователь» как один ко многим, один пользователь ко многим задачам, вторая это связь с моделью «Проект» указывает на проект к которому относиться данная сущность связь один ко многим. Из представленной диаграммы моделей сущностей получаем физическую диаграмму.

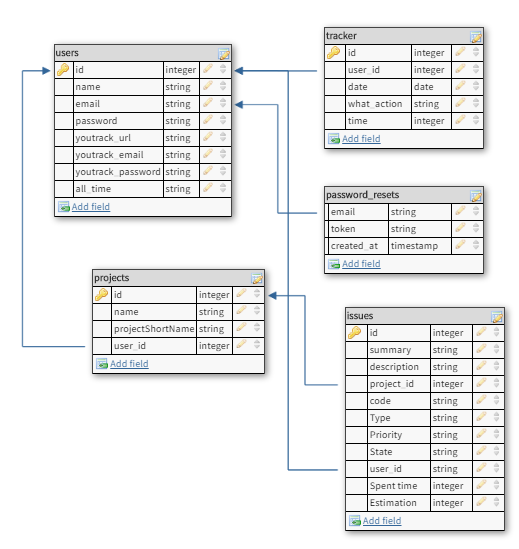


Рисунок 2.2 – Физическая диаграмма моделей сущностей

На рисунке 2.2 представлена физическая диаграмма, реализуема в СУБД в ней мы видим отличия от ER – диаграммы в том что добавились особенности реализации такие как допустимые типы и наименования полей и таблиц, а так же внешние ключи для связей таблиц.

## 2.4 Программная структура проекта

Проект реализован на фреймворке «Laravel» с использованием паттерна MVC, рассмотрим основную структуру проекта.

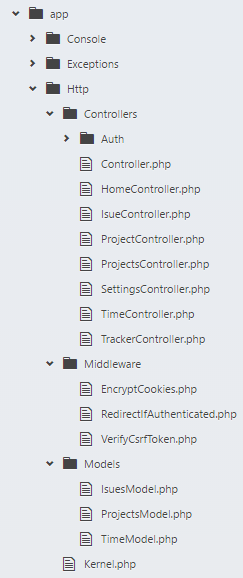


Рисунок 2.3 – Структура фалов контроллеров и моделей

На рисунке 2.3 представлена основная структура файлов контроллеров и моделей проекта. Как мы видим, в пространстве имен App/Http/Contollers находятся контроллеры, они отвечают за обработку HTTP запросов, а так же за получение данных из базы посредством моделей находящихся App/Http/Models. На основе данной структуры составим UML диаграмму классов.

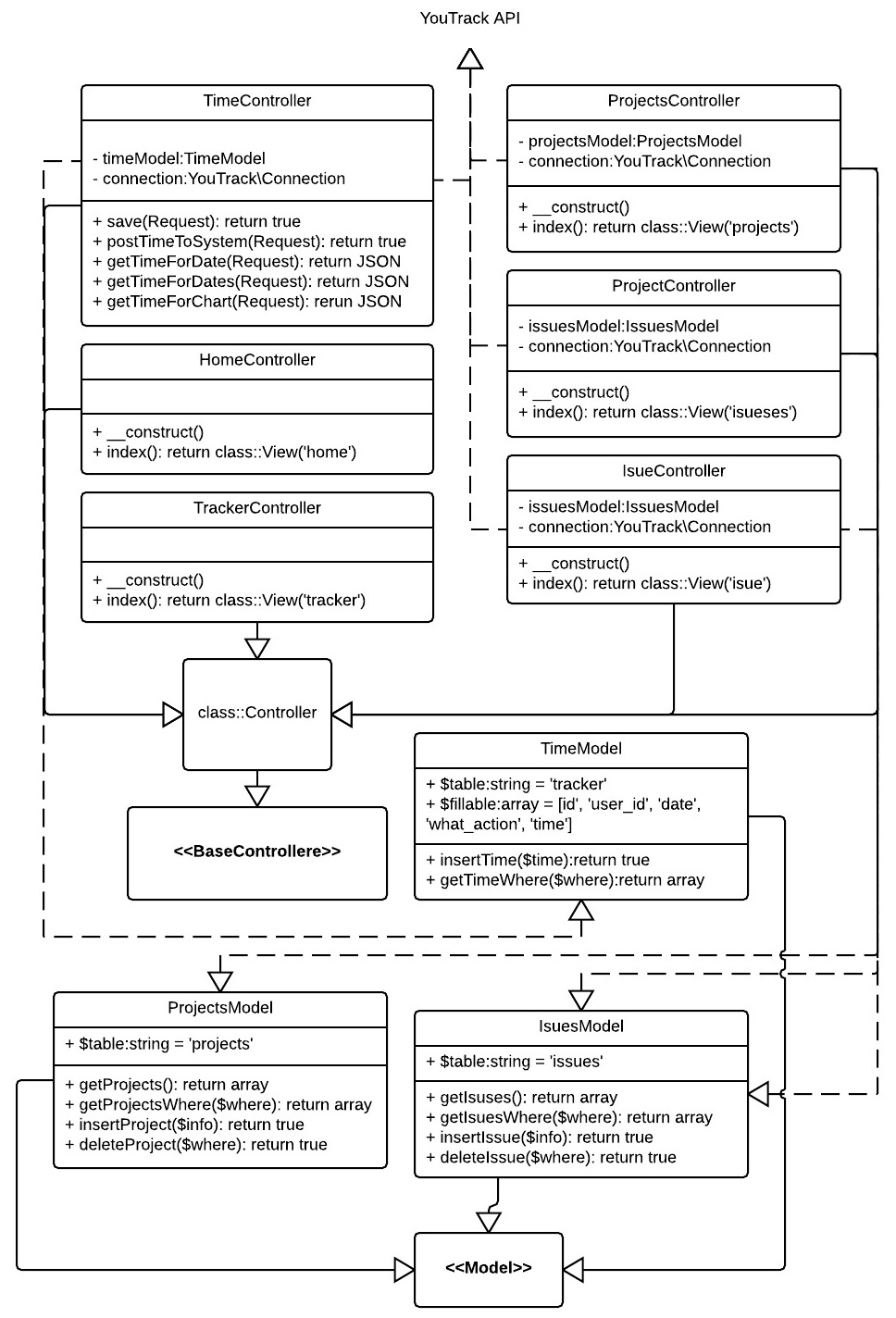


Рисунок 2.4 – Структура фалов контроллеров и моделей

## 2.5 Use case диаграмма использования

Для системы была составлена диаграмма использования для всех типов пользователей.

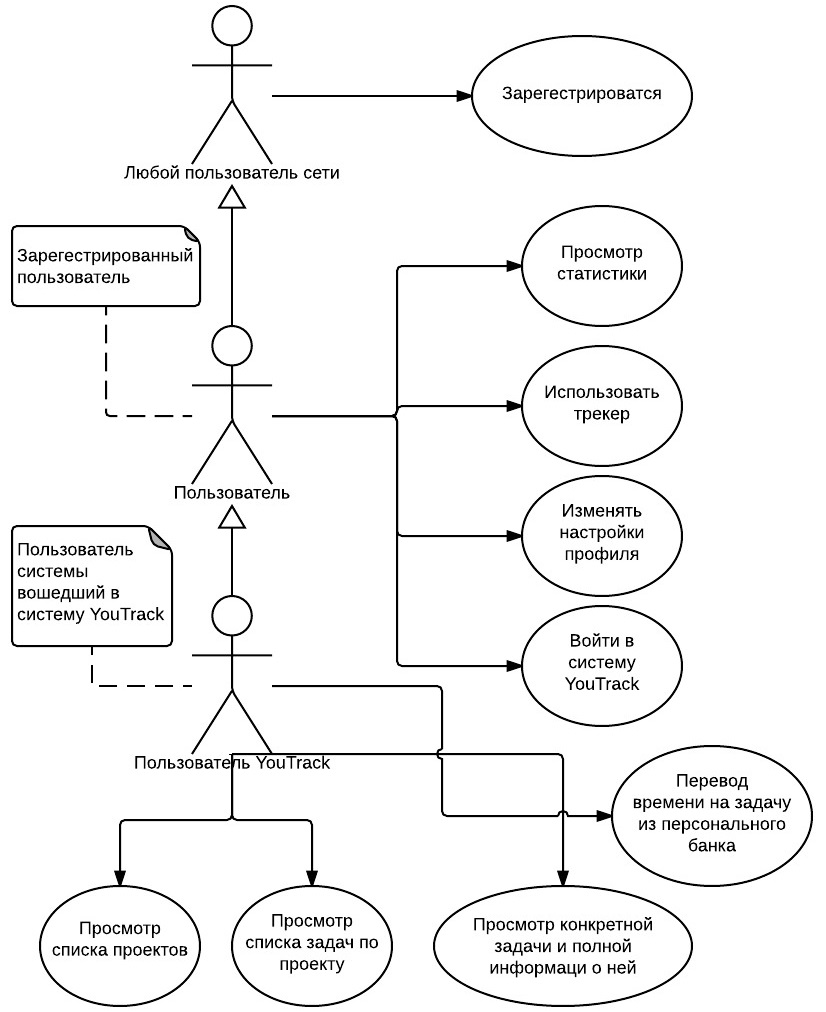


Рисунок 2.5 – Use case диаграмма использования