Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет инженерно–экономический

Кафедра экономической информатики

*К защите допустить:*

\_\_\_\_\_\_\_\_\_\_В.Н. Комличенко

**ПОЯСНИТЕЛЬНАЯ** **ЗАПИСКА**

к курсовому проекту

на тему

**АВТОМАТИЗАЦИЯ ПРОЦЕССА УПРАВЛЕНИЯ РАБОЧИМ ВРЕМЕНЕМ СОТРУДНИКОВ IT-КОМПАНИИ**

**БГУИР КП 1-40 01 02-08 ПЗ**

Студент А.Ю.Мартыненко

Руководитель А.Г.Хмелёв

Минск 2016

[ВВЕДЕНИЕ 4](#_Toc469451500)

[1. Исследование организации процесса управления рабочим временем 6](#_Toc469451501)

[1.1 Сравнительный (критический) анализ существующих аналогов систем управления рабочим временем 6](#_Toc469451502)

[1.2 Общие понятия и термины 14](#_Toc469451503)

[2. Анализ процесса управления рабочим временем сотрудников it-компании 20](#_Toc469451504)

[2.1 Описание процесса руководства и проверки курсового проекта 20](#_Toc469451505)

[2.2 Анализ процесса управления рабочим временем сотрудников IT-компании 25](#_Toc469451506)

[3. Разработка автоматизированной системы управления рабочим временем сотрудников it-компании 28](#_Toc469451507)

[3.1 Постановка задачи и описание вариантов использования системы 28](#_Toc469451508)

[3.2 Выбор и обоснование архитектурных решений 32](#_Toc469451509)

[3.3 Функциональное моделирование процесса управления рабочим временем 39](#_Toc469451510)

[3.4 Информационное моделирование системы управления рабочим временем 41](#_Toc469451511)

[3.5 Разработка моделей представления системы управления рабочим временем 44](#_Toc469451512)

[3.6 Описание алгоритмов программных модулей системы управления рабочим временем 49](#_Toc469451513)

[3.7 Результаты проектирования системы проверки курсового проекта 49](#_Toc469451514)

[Заключение 60](#_Toc469451515)

[Список использованных источников 61](#_Toc469451516)

[Приложение А](#_Toc469451517) [(обязательное)](#_Toc469451518) [Функциональная модель «TO BE»  
 (к пункту 3.3) 62](#_Toc469451519)

[Приложение Б](#_Toc469451520) [(обязательное)](#_Toc469451521) [Cхема функции (к пункту 3.6) 64](#_Toc469451522)

[Приложение В](#_Toc469451523) [(обязательное)](#_Toc469451524) [Листинг кода основных элементов   
 программы 65](#_Toc469451525)

[Приложение Г](#_Toc469451526) [(обязательное)](#_Toc469451527) [Листинг скрипта создания базы данных 77](#_Toc469451528)

ВВЕДЕНИЕ

В наше время для огромного количества клиентов различных сервисов [разработка приложений для мобильных](http://woxapp.com/ios-apps-development/) телефонов и других гаджетов играет большое значение. Именно с их помощью можно получать возможности пользоваться теми или иными услугами, делать покупки без особых затрат усилий и времени.

Постоянный рост конкуренции в сферах среднего и малого бизнеса принуждает руководителей компании все чаще задумываться о том, чтобы применить доступные современные ИТ-решения для оптимизации существующих бизнес-процессов и увеличения собственной конкурентоспособности. В последние время эта тенденция заставила разработчиков софта пересмотреть свой привычный подход в реализации продуктов, и выйти на новый этап поддержки бизнес-процессов компаний с применением доступных мобильных технологий.

В настоящее время мобильные технологии все шире применяются в управлении и автоматизации производственных процессов. Это стали учитывать и разработчики мобильных устройств, беспроводных информационных сетей, оборудования для информационной инфраструктуры, а также соответствующего ПО.

Невозможно недооценить [роли мобильных приложений в эффективности работы](https://www.wrike.com/ru/library/ebooks/mobile-productivity-report-2016/). Так, около 70% пользователей специальными приложениями для управления эффективностью распределения рабочего времени отметили, что использование мобильных устройств отрицательно не сказалось на балансе их личной и профессиональной жизни и, напротив, даже улучшило его. 45% тех, кто часто работает с мобильных устройств, утверждают, что для сохранения своей продуктивности они используют 5 и более приложений.

Процесс управления временем необходим в современном мире. От тайм-менеджмента зависит работа сотрудников, отделов, и даже компаний. Тайм-менеджмент позволяет контролировать время и оценивать его в течении некоторого периода. Главной особенностью тайм-менеджмента в современном мире является то, что он не может быть применён единожды для получения эффекта. Управлять временем необходимо регулярно, для того, чтобы человек мог ценить свободное время, определять возможности, не тратить драгоценные минуты на прочтение мусорной информации в интернете. Поэтому временем нужно управлять в течение жизни.

Данная тема актуальна как со стороны рассмотрения технологий, так и со стороны рассмотрения темы курсового проекта. Ведь при объединении мобильных технологий и тайм-менеджмента современный человек может контролировать себя, определять задачи на некоторое время и отмечать процесс их выполнения, пользоваться различными методиками контроля времени, и всё это можно сделать без использования дополнительных подручных средств, только с помощью мобильного телефона. Данный проект позволит с минимальными затратами максимизировать эффективность тайм-менеджмента.

Таким образом, объектом исследования явился процесс планирования и учета рабочего времени в компании.

Предмет исследования – выявление условий и путей повышения эффективности учета рабочего времени.

Целью данного курсового проекта является помощь в учете рабочего времени сотрудников и упрощения контроля временных затрат на различные задачи.

Перечень задач курсового проекта, которые необходимо выполнить для достижения поставленной цели:

* исследовать процесс планирования и учета рабочего времени;
* провести анализ существующих аналогов разрабатываемой системы;
* провести анализ применяемых инструментальных и технических средств;
* охарактеризовать основные процессы планирования и учета рабочего времени;
* определить базовые принципы, которыми необходимо руководствоваться при проектировании системы;
* сформулировать архитектурное решение для разработки системы;
* выбрать и обосновать технические, технологические и программные средства, которые будут использованы при проектировании системы;
* выполнить программную реализацию системы;
* провести тестирование программного продукта.

Таким образом, были определены цели и задачи курсового проекта, определены объект и предмет разработанной системы, а также обозначена актуальность данной темы в современном мире.

# ИССЛЕДОВАНИЕ ОРГАНИЗАЦИИ ПРОЦЕССА УПРАВЛЕНИЯ РАБОЧИМ ВРЕМЕНЕМ

* 1. Сравнительный (критический) анализ существующих аналогов систем управления рабочим временем

Одним из самых дорогих и невосполнимых ресурсов, которым обладает человечество, является время. В среднем человек живёт 66,29 лет в мире, а данная цифра эквивалентна 24212 дням или 34865280 минутам. Большую часть времени – почти треть – человек тратит на сон. 30% оставшейся части среднестатистический человек тратит на бытовые проблемы. Поэтому на реализацию в работе, в учёбе, в семье уходит только лишь половина от всего отведённого нам времени – 49%. Поэтому очень важно научится правильно реорганизовывать его для того, чтобы успеть сделать больше. А для этого необходимо понимать основы управления временем.

Управление временем, или тайм-менеджмент, - это технология организации времени и повышения эффективности его использования [1]. Если рассматривать данное понятие глобально, то можно отметить, что управление временем в первую очередь является стилем жизни, или можно также сказать философией ценностей в быстром потоке жизни, а не набор разработанных техник. Применив определённый способ для решения проблем организации времени, результат может быть получен в течении нескольких месяцев, а иногда и лет.

В тайм-менеджменте обычно выделяют следующие процессы:

* постановка цели;
* планирование и расстановка приоритетов;
* разработка плана достижения поставленных задач;
* реализация разработанного плана;
* контроль достижения цели и выполнения разработанных планов.

Данные процессы представлены на рисунке 1.1 в виде диаграммы. Процессы характеризуют последовательность действий, которую необходимо выполнить для оптимального контроля своего времени. Также можно отметить, что данные процессы могут и даже будут повторяться, так как тайм-менеджмент не проводится единожды и процессы в течении времени могут меняться.

Рисунок 1.1 – Процессы управления временем

В начале своего появления тайм-менеджмент касалось только трудовой сферы человеческой жизни, затрагивая вопросы бизнеса. Однако по мере развития данный термин стал стремительно расширяться и включать различные аспекты человеческой жизни. Сегодня управление временем является необходимой составляющей развития абсолютно любого проекта, так как служит определяющим фактором при расчёте его масштаба и времени, требуемого для его реализации [2].

Как уже было сказано, тайм-менеджмент необходим для организации бизнеса в современном мире. Он позволяет определить, какое количество задач можно выполнить и какое количество времени потратить для их реализации. Рассмотрим данный вопрос более конкретно.

Рабочее время — время, затрачиваемое на выполнение работы, время активного пребывания на работе, посвящённое выполнению прямых служебных обязанностей. Реально рабочим временем считается время, отмеченное в документе учёта времени нахождения на работе [3].

Если не проводить анализ рабочего времени сотрудника, можно прийти к возникновению дефицита рабочего времени.

Дефицит рабочего времени — нехватка временного ресурса, вызванная неправильной организацией работником своей деятельности, либо неграмотной организацией деятельности руководством, что приводит к спешке, затягиванию выполнения работ, заданий, некачественной работе, потерям в производстве, браку, что в конечном итоге существенно влияет на эффективность и результаты работы всего предприятия [4].

Наиболее характерными причинами дефицита времени являются следующие:

* бесплановость работы;
* несоответствие работника и занимаемой им должности;
* неадекватная оценка своих способностей, скорости выполнения задачи, результативности;
* отсутствие личной миссии работника;
* неумение контролировать свои потребности (в питании, мыслях, выражении эмоций);
* слабая мотивация труда.

Для анализа деятельности сотрудника в рабочее время обычно применяются техники и методики тайм-менеджмента. Раньше на практике чаще всего встречались методы расчёта различных коэффициентов, позволяющие определить затрачиваемое время сотрудника на определённую деятельность. Рассмотрим их более подробно.

Коэффициент экстенсивного использования рабочего времени используется для подсчёта рабочего времени сотрудника. В простейшем случае коэффициент экстенсивного использования рабочего времени рассчитывается по формуле:

(1.1)

где Р – регламентированные и нерегламентированные перерывы в работе, мин;

Ф – общее время работы, или фонд работы, мин.

Коэффициент потерь рабочего времени, которые зависят от работников, позволяет получить информацию о том, насколько ответственно работник подходил к своим обязанностям и выполнению рабочих задач. Данный коэффициент рассчитывается по формуле:

(1.2)

где Пз – потери рабочего времени, зависящие от работника, мин.

Коэффициент потерь рабочего времени, которые обусловлены организационно-техническими причинами на месте работы, позволяет получить информацию о том, насколько часто сотрудник отвлекается на нерабочую обстановку, не зависящую от него. Обычно данный коэффициент зависит от деятельности всего рабочего коллектива. Он рассчитывается по формуле

(1.3)

где По – потери рабочего времени, не зависящие от работника, мин.

Коэффициент затрат рабочего времени на личный отдых или потребности человека (обед, гигиена, производственная гимнастика и так далее) вычисляется по формуле

(1.4)

где ОЛ – затраты времени работника на личные нужды, мин.

Стоит отметить, что данными показателями довольно редко пользуются в современном мире, так как существует ряд техник и методик, которые точнее и эффективнее рассчитывают время сотрудников за работой, ведь в век высоких технологий очень сложно выявить отвлечение от работы. Так что можно с уверенностью сказать, что данные оценки времени устарели.

На сегодняшний момент существует около 3 000 различных методик управления временем. Поэтому будут рассмотрены наиболее устоявшиеся методики, которые используются по всему миру.

Одним из популярнейших методик управления временем является метод Эйзенхауэра, подразумевающий построение матрицы Эйзенхауэра. Матрицей Эйзенхауэра сегодня пользуются не только для планирования рабочего времени, но и для повседневных целей. Основоположником этой матрицы является тридцать четвёртый президент США Дуайт Дэвид Эйзенхауэр. Эйзенхауэр занимал высокую должность, поэтому должен был оптимизировать свой рабочий график и список выполняемых задач. Итогом его изысканий и стала рассматриваемая нами матрица.

Матрица Эйзенхауэра представляет собой четыре квадранта, основанием которых служат две оси — это ось важности (по горизонтали) и ось срочности (по вертикали). В итоге получается, что каждый квадрант отличается своими качественными показателями. В каждый из квадрантов записываются все задачи и дела, благодаря чему образуется предельно ясная и объективная картина того, чем следует заняться в первую очередь, чем – во вторую, а чем вообще заниматься не стоит [6]. Данная матрица представлена на рисунке 1.2.



Рисунок 1.2 – Матрица Эйзенхауэра

Вторым по популярности методом управления временем является принцип Парето. Закон Парето (принцип Парето), или правило 80/20 – один из наиболее распространенных способов оценки эффективности какой-либо деятельности. Его суть заключается в том, что 20% усилий дают 80% результата, а остальные 80% усилий реализуют лишь 20% результата. Таким образом, можно понять, что, выбрав те оптимальные ресурсы, которые дают наибольший эффект, можно достичь высоких результатов малыми издержками. В то же время последующие усилия будут ненужными и неэффективными.

Принцип Парето оказала очень большое влияние на успех многих людей: бизнесменов, изобретателей, инвесторов и так далее. Осознание этой простой истины привело к изменениям в нашем мире этими людьми [7]. Закон Парето представлен на рисунке 1.3.

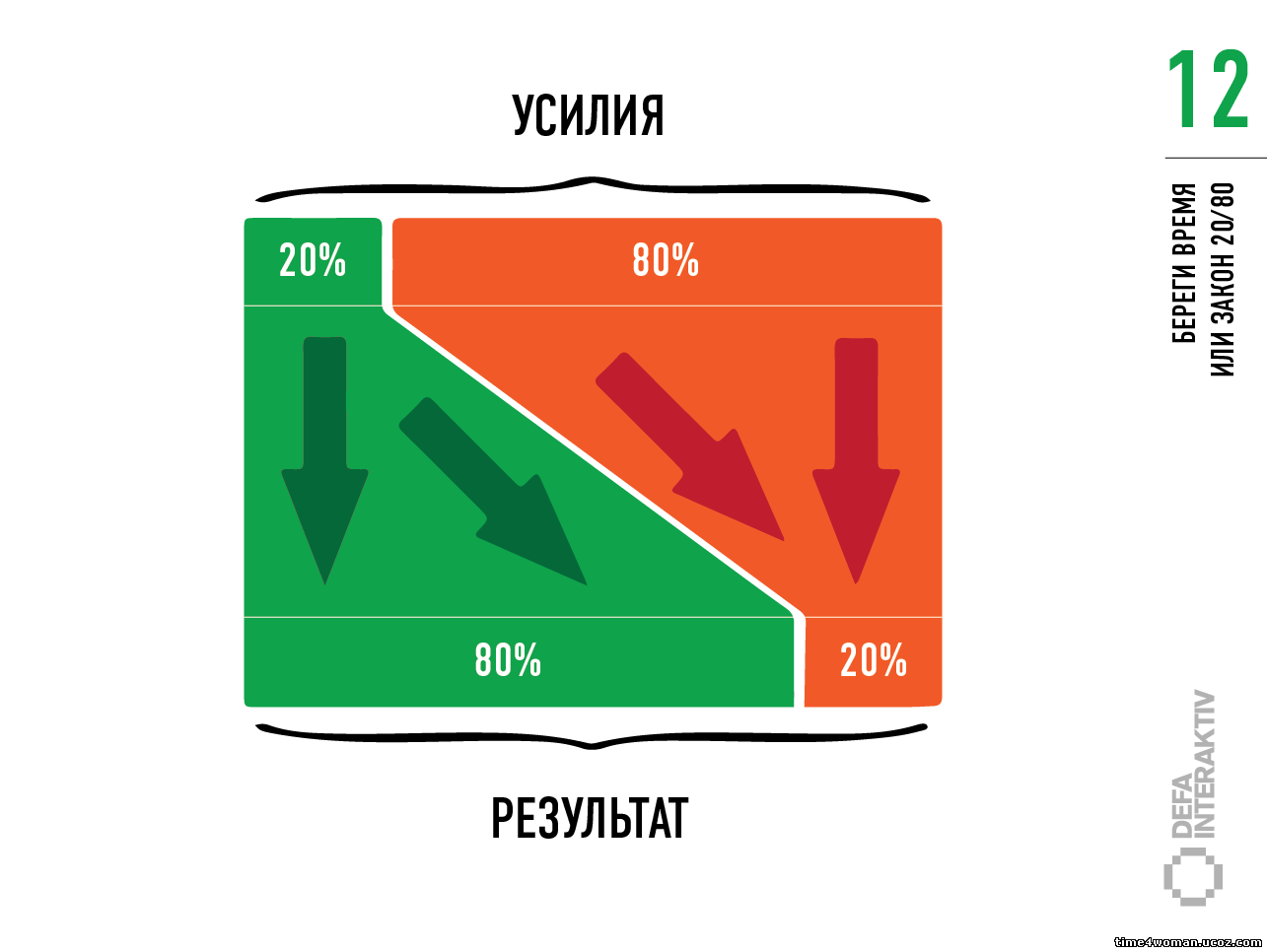


Рисунок 1.3 – Принцип Парето

Диаграмма Ганта также считается одним из наиболее удобных и популярных способов графического представления времени выполнения задач, является наглядным и удобным инструментом для управления проектом. Первый формат диаграммы был разработан Генри Л. Гантом в 1910 году.

Диаграмма представляет собой совокупность полос, ориентированных вдоль оси времени. Каждая полоса – это отдельная задача в составе проекта, концы представляют начало и завершение работы, протяжённость является длительностью работы. Вертикальная ось – перечень задач. Кроме того, на диаграмме могут быть отмечены совокупные задачи, проценты завершения, указатели последовательности и зависимости работ, метки ключевых моментов (вехи), метка текущего момента времени «Cегодня» и другое. Также с помощью стрелок могут быть отображены взаимосвязи задач друг с другом. Диаграмма предоставляет возможность наглядно представлять как последовательные шаги выполнения проекта, так и задачи, требующие одновременного выполнения [8]. Диаграмма Ганта представлена на рисунке 1.4.

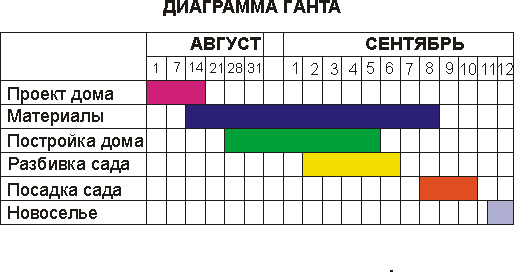


Рисунок 1.4 – Диаграмма Ганта

Хронометраж - метод изучения затрат времени с помощью фиксации и замеров продолжительности выполняемых действий [4]. Хронометраж позволяет провести «аудит» и «инвентаризацию» времени, выявить «поглотителей времени». Для того чтобы провести хронометраж, рекомендуется записывать все свои дела по созданным вами категориям с точностью до 5-10 минут в течение хотя бы двух недель. Данная методика является наиболее тяжёлой, так как перерывы между записями довольно маленькие. Однако отслеживание таким образом своё рабочее время, можно довольно быстро оптимизировать работу. Пример таблицы хронометража представлена на рисунке 1.5.

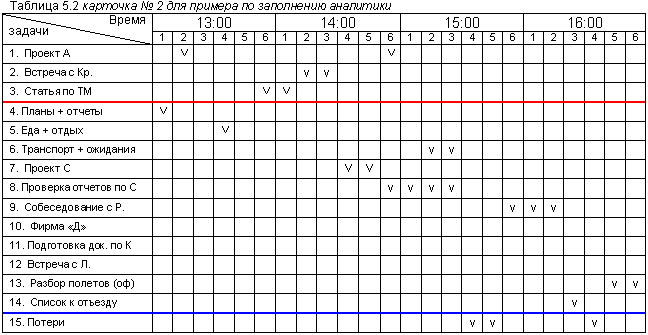


Рисунок 1.5 – Таблица хронометража

Наиболее традиционной методикой является создание списков дел. Данный способ используют даже те люди, которые и не слышали про тайм-менеджмент. Данная методика отличается лёгкостью управления задачами и быстротой их решения, однако она не всегда эффективна в современном мире.

Следует отметить, что для всех представленных техник необходимо правильно определить цель. Наиболее подходящая для этого методика – технология постановки и формулировки целей SMART. Данный акроним означает «умная» цель, что обозначает, какой должна быть настоящая цель. Для этого используются пять прилагательных:

* Specific (конкретность, то есть цель должна быть максимально ясной и конкретной);
* Measurable (измеримость, то есть должны определяться числовыми показателями);
* Attainable (достижимость, то есть должны быть достижимы с точки зрения внешних и внутренних факторов);
* Relevant (релевантность, то есть должны соотносится с другими, более важными, целями, и быть стратегическими);
* Time-bounded (определенность во времени, то есть должна подходить под конкретные сроки) [9].

Расшифровка аббревиатуры SMART цели представлена на рисунке 1.6.



Рисунок 1.6 – SMART цели

Для людей, которые следуют представленным технологиям и методикам, через некоторое время придём осознание полного контроля своего времени. Однако некоторые люди не могут работать и жить по чётко-структурированному плану. Для таких людей существуют ряд советов по проведению своего времени, придерживаясь которых человек сможет эффективно контролировать, что он делает, тем самым управлять временем [9]. Рассмотрим советы в виде диаграммы, изображённой на рисунке 1.7.

Рисунок 1.7 – Советы по управлению временем

Таким образом, если придерживаться определённых советов для управления временем, а также полагаться на особые технологии и методики тайм-менеджмента, можно значительно повысить эффективность своей работы, тем самым больше проводить время с семьёй, заниматься личными делами и другое.

* 1. Общие понятия и термины

Сегодня современные технологии могут предложить нам множество возможностей для более продуктивного управления временем. Рассмотрим некоторые из них.

Для начала рассмотрим приложение Timesheet. [Timesheet](https://play.google.com/store/apps/details?id=com.llamalab.timesheet.free) —бесплатное приложение для Android, позволяющее создавать голосовые заметки, контролировать и планировать свое рабочее и свободное время (рисунок 1.8). Также предусмотрена возможность расчета оплаты работы (исходя из затраченного на неё времени) и создания соответствующих отчетов в Excel, PDF и QuickBooks.

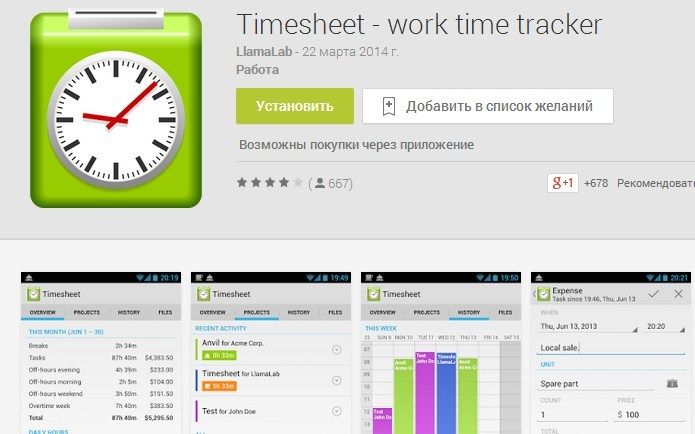


Рисунок 1.8 – Интерфейс приложения Timesheet

Workflow — это легко настраиваемое приложение, которое сокращает ваши временные затраты на выполнение сложных ежедневных задач (рисунок 1.9).

С помощью Workflow вы можете создать уникальные ярлыки на «рабочем столе» вашего телефона или планшета, которые выполняют серии задач одним кликом. Что бы вас не интересовало — автоматическое «подтягивание» маршрута к месту вашей следующей встречи из Google Maps или резервное копирование всех ваших фотографий в Dropbox’е — у вас появится возможность разрабатывать свои собственные приложения, которые будут отвечать за любые необходимые функции.

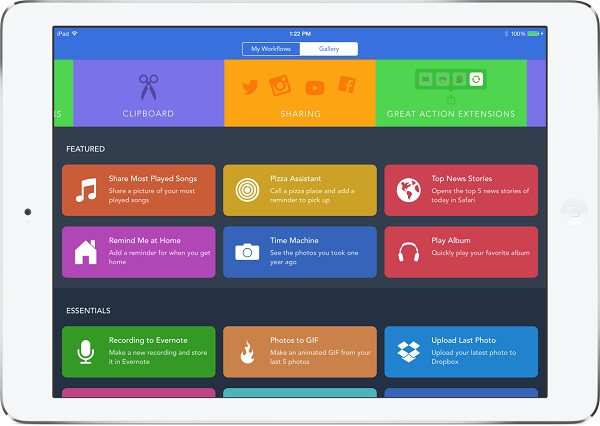


Рисунок 1.9 – Интерфейс приложения Workflow

Workflow может похвастаться более чем 200 действий и интеграций с такими приложениями, как Facebook, Uber и YouTube. Этот тайм-менеджер доступен на iOS.

Workflow помогает вам сэкономить время за счет автоматизации сложных задач, которые вы регулярно выполняете на мобильном устройстве.

Workflow отличается простотой и скоростью работы, ведь он позволяет вам быстро создавать новые приложения и ярлыки при помощи обычных «drag-and-drop» команд.

Workflow универсален и может работать внутри таких приложений, как Safari и Evernote.

Существуе также хорошая аналогия разработанному проекту – Google Now. Google Now — это интеллектуальный голосовой помощник, который обрабатывает сложные запросы и использует ваше поведение, чтобы предугадывать, какая информация вам понадобится, еще до того, как вы попросите об этом (рисунок 1.10).

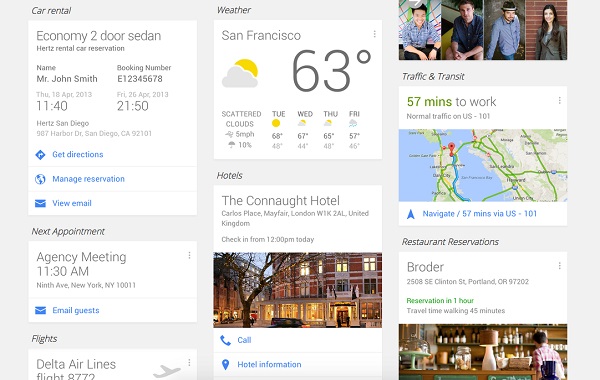


Рисунок 1.10 – Интерфейс приложения Google Now

Благодаря доступу к обширной сети данных Google Now может предоставлять вам сведения, которые посчитает релевантными для вас. От результатов спортивных состязаний и близлежащих ресторанов до наличия пробок по дороге на работу, Google Now высвобождает ваше время, заботясь о всех ваших ежедневных мелочах.

Google Now также полностью интегрирован в экосистему приложений Google, что позволяет вам назначать мероприятия в Google Календаре, вносить заметки и многое другое. Этот онлайн-ассистент совместим с операционными системами iOS и Android.

Google Now дает вам больше времени, чтобы сосредоточиться на своей работе, ведь он предоставляет вам нужную информацию, не заставляя вас копаться в поисковой выдаче.

Google Now всегда готов помочь, хотите вы зарезервировать столик в ресторане или пытаетесь вспомнить, где припарковали свой автомобиль.

Приложение автоматически адаптируется к вашему поведению и обеспечивает вам уникальный и полностью настраиваемый опыт, который можно изменить в любой момент.

[Toggl](https://www.toggl.com/) — сервис, позволяющий контролировать и планировать рабочее время конкретного пользователя или целой команды (рисунок 1.11). Добавив к списку команды своих сотрудников, вы сможете отслеживать время, затраченное ими на выполнение поставленных задач. В программе можно создавать общие и индивидуальные отчеты.

Toggl поддерживает интеграцию с различными популярными сервисами (Basecamp, JIRA, Teamweek, Google Calendar и т.д.). Приложение можно установить на ПК, мобильные устройства или работать прямо в браузере.

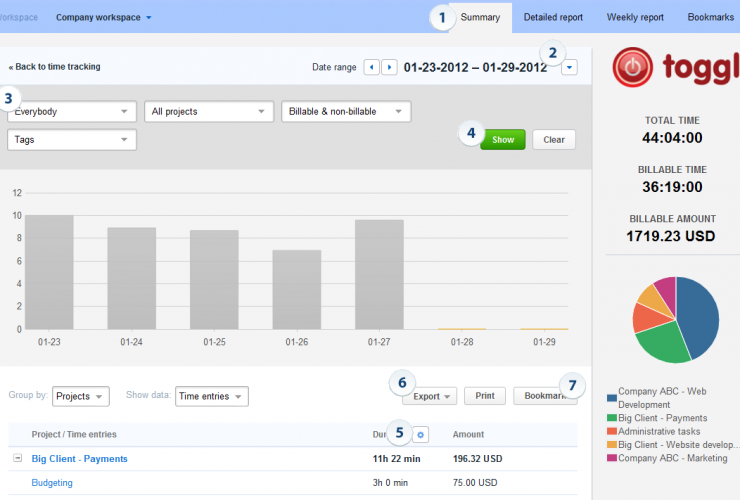


Рисунок 1.11 – Интерфейс приложения Toggl

С помощью данного приложения можно эффективно управлять временем. Например, здесь определены такие функции, как создание задачи, определение продолжительности, хронометраж, вычисление различных степеней срочности задчи, построение графиков и многое другое. Данное приложение является отличным продуктом для новичков, которые только посвящаются в тайм-мененджмент. Также в интернете существует множество обзоров работы данного приложения, для того, чтобы пользователь при возникновении непредвиденных ситуаций мог спокойно найти ошибку с помощью гайдов. Также следует отметить, что Тoggl интегрируется в несколько десятков приложений, в том числе в Браузер Google Chrome и популярную программу тайм-менеджмента Todoist. Существуют решения для iPhone и Android.

Сравнительная характеристика данных приложений с точки зрения ключевых аспектов представлена в таблице 1.1.

Таблица 1.1 - Сравнительная характеристика приложений по управлению рабочим временем сотрудников ИТ компании

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Timesheet | Workflow | Google Now | [Toggl](https://www.toggl.com/) |
| Контроль рабочего времени | + | + | + | + |
| Контроль свободного времени | + | - | + | - |
| Наличие десктопной версии | - | - | - | + |

Недостатки данных мобильных приложений при рассмотрении ИТ компании в том, что они направлены на более широкий спектр задач, не всегда связанных с конкретными рабочими задачами. Многие из них рассчитаны на планирование своего дня, поездок и личных дел. Таким образом, деловая сфера жизни человека в этих приложениях пересекается с личной, что создает трудности в разделении этих областей. Также сотрудник может слишком часто отвлекаться на нерабочие дела, так как в приоритете они могут оказаться выше. Такой тайм-менеджмент может оказаться неэффективным.

Таким образом, в данной главе были рассмотрены основные понятия тайм-менеджмента, представлены основные методики управления временем, а также предложены советы по управлению временем для людей, которые не могут пользоваться долго определённой методикой. Далее были рассмотрены основные конкуренты разработанного продукта на рынке, что в последствии поможет определить ценность разработанного приложения.

# АНАЛИЗ ПРОЦЕССА УПРАВЛЕНИЯ РАБОЧИМ ВРЕМЕНЕМ СОТРУДНИКОВ IT-КОМПАНИИ

Процесс управления временем сотрудников любой ИТ компании довольно сложен, так как подразумевается, что сотрудник должен управлять своим временем самостоятельно, а значит всё ложится на плечи рабочего. Поэтому необходимо рассмотреть данный процесс более детально и определить узки места, которые должны быть учтены при проектировании системы по управлению рабочим временем сотрудника.

* 1. Описание процесса руководства и проверки курсового проекта

В современных компаниях персонал выделяет огромное количество времени на планирование рабочего дня. Сотрудники планируют его в зависимости от количества задач, которые необходимо выполнить к определённому сроку. Однако само планирование задач занимает определённое рабочее время. Рассмотрим этот процесс при помощи нотации IDEF0.



Рисунок 2.1 – Контекстная диаграмма процесса управления временем

На рисунке 2.1 изображена контекстная диаграмма процесса управления временем. Здесь чётко просматривается, что на процесс управления временем сотрудников влияют такие составляющие, как нормативная документация, время на выполнение задания, а также расписание сотрудника на день.

Далее необходимо определить, какие процессы определяются в течении рабочего времени сотрудника. Декомпозиция, представленная на рисунке 2.2, определяет собой последовательный процесс от подготовки рабочего места к выполнению задания до подготовки отчёта по его выполнению. На диаграмме видно, что входной параметр «Предоставленное рабочее время» постепенно преобразуется в выходной «Спланированное время». Также можно отметить, что в процессе явно просматривается цикл, отходящий от предоставленного результата и переходящий к подготовке к работе.



Рисунок 2.2 – Декомпозиция процесса управления временем сотрудника

Рассмотрим декомпозиции представленных процессов. На рисунке 2.3 представлена декомпозиция процесса подготовки к работе сотрудника. Если рассмотреть данный процесс, то можно выделить, что в действительности процесс подготовки к работе разделяется на четыре этапа: получить задание, ознакомится с правилами выполнения, проинструктироваться, непосредственно подготовить рабочее место к выполнению задания. Все эти процессы напрямую связаны со временем исполнения задания.



Рисунок 2.3 – Декомпозиция процесса подготовки сотрудника к работе

На рисунке 2.3 видно, что после того, как сотрудник прошёл через все этапы подготовки, у него осталось значительно меньше времени на выполнение задания, чем было при его получении. Поэтому на данной диаграмме можно увидеть узкое место, которое говорит о том, что время тратится не на то, что действительно важно для работы, а на определённые действия, которые к работе относятся косвенным образом.



Рисунок 2.4 – Декомпозиция процесса выполнения работы сотрудником

На рисунке 2.4 изображён этапы процесса выполнения работы. Данный процесс определён с помощью пяти последовательных этапов: непосредственно выполнения задания, прерывание на отдых, исправление недостатков, которые были допущены при выполнении работы, перерыв по техническим причинам и завершение работы. Как видно на рисунке, процесс довольно долгий и требует много времени.



Рисунок 2.5 – Декомпозиция процесса предоставления результата

Как видно на рисунке 2.5, процесс предоставления результата, как и предыдущие процессы, происходит с большими затратами времени. Делится рассматриваемый процесс на следующие этапы: приход с результатами к вышестоящему лицу, предоставление результатов, оценка работы, подведение итогов. Входные и выходные данные, как и в предыдущих процессах, определяются оставшимся временем после определённого этапа.



Рисунок 2.6 – Декомпозиция процесса формирования отчёта по выполненной работе

На представленном рисунке 2.6 можно увидеть, что последний процесс также определён этапами, которые требуют большого количества временных ресурсов. Также на диаграмме видно, что сотрудник может определить время, которое он тратит на выполнение задания, только когда выполнит это задание и предоставит отчёт начальнику. Поэтому для того, чтобы времени сотрудник тратил на всё меньше и успевал делать больше работы за определённого рабочее расписание, необходимо автоматизировать данный критерий.

* 1. Анализ процесса управления рабочим временем сотрудников IT-компании

Те, кто добился успеха в своей жизни, много времени посвящают планированию. Ежедневное планирование просто необходимо для повышения производительности и эффективного управления временем.

Правило 6 «П» гласит: правильное предварительное планирование предотвращает плохие показатели.

Одной из основных проблем в процессе управления рабочим временем сотрудников ИТ компании является сложность самостоятельного мониторинга и контролирования своей деятельности. Ведь время должно быть разумно распределено с учетом приоритетов поставленных задач, их срочности, важности, необходимости в особых условиях и т.д. Схема разделения задач по основным категориям представлена на рисунке 2.7. Поэтому с целью выполнения поставленных задач в нужный срок и в качественном виде сотрудники должны вести планирование своего времени.

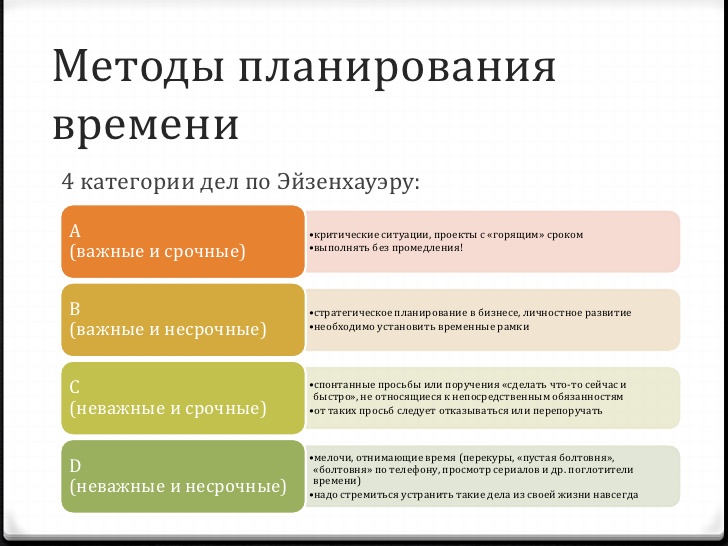


Рисунок 2.7 – Категории дел

Данная необходимость решает задачи распределение трудовых и временных ресурсов, позволяет сотрудникам заниматься самоконтролем и обеспечивает то, что задачи никогда не будут забыты и задвинуты в дальний ящик.

Также еще одна из целей – упрощение деятельности руководителя, снятие с него нагрузки, связанной с напоминанием о необходимости выполнения той или иной задачи. Все отношения между руководителем и сотрудником значительно упрощаются.

Все перечисленные задачи можно решить путем автоматизации данного процесса. Для этого можно спроектировать, а в дальнейшем создать приложение по автоматизации деятельности сотрудников и ведения планирования и контроля рабочего времени. Оно будет подчиняться правилам и методикам тайм-менеджмента.

Для того, чтобы данное приложение по-настоящему стало «карманным» и было все время под рукой, было принято решение о его мобильной версии.

Разработанное приложение должно подчиняться основным задачам тайм-менеджмента:

1. Постоянный анализ распорядка дня, а также дел, которые за день выполняются, и оптимизация этого распорядка;
2. Самоконтроль в процессе работы – когда и что приходится делать;
3. Изучение и применение на практике техники рационального использования времени;
4. Умение отказываться от деятельности, которая не приносит значимых результатов и занимает много времени;
5. Умение планировать время рационально, уделяя повышенное внимание наиболее важным и значимым делам;
6. Организация своего личного времени таким образом, чтобы отдых был полноценным.

Разработанное приложение можно будет использовать не только в ИТ компаниях, но и в других сферах общества, будь то волонтерские организации, в которых над какими-то проектами смогут трудиться несколько волонтеров, или для трекинга времени работы студентов над курсовыми, но использовать это в паре с системой контроля версий.

# РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ РАБОЧИМ ВРЕМЕНЕМ СОТРУДНИКОВ IT-КОМПАНИИ

* 1. Постановка задачи и описание вариантов использования системы

Задачей курсового проекта является разработка системы управления рабочим временем сотрудников IT-компании, позволяющая упростить контроль за учетом времени уходящим на определенные задачи. Данная система предоставит возможность отслеживать для менеджеров и директоров задачи сотрудников и так же время, затраченное на задачи. Данный подход позволит объективно оценивать результаты сотрудника и в любой момент посмотреть историю фиксированных точек этапа выполнения конкретной задачи.

На основе проведенного анализа были выделены следующие требования, которым должна соответствовать система:

* обеспечить возможность работы приложения таким образом, чтобы у сотрудников и менеджеров не было проблем с использованием;
* обеспечить отказоустойчивость системы для управления временем;
* обеспечить безопасность работы приложения, т.е. чтобы сотрудники не могли получить доступ просмотру информации, к которой у них нет доступа;
* обеспечить универсальный алгоритм трекинга времени как для сотрудников и гостей компании, таких как практиканты, временные сотрудники, так и для управляющего звена компании.

На рисунке 3.1 определены варианты использования системы с учетом решаемых задач и поставленных требований. В диаграмме отображены 4 роли:

* Неавторизованный пользователь
* Пользователь
* Работник
* Администратор

Неавторизованный пользователь – это лицо, которое зашло в приложение, но ни как, не авторизовало себя в системе. Данный тип роли отображает пользователя с очень ограниченными правами в приложении.

Пользователь – это лицо которое авторизовалось, либо зарегистрировалось в системе и может пользоваться полноценно приложением с уровнем прав, что позволяет программа. Данный тип роли подразумевает, что пользователь может не быть сотрудником компании, но соглашается с условиями пользования приложением и использованием его личной информации приложением.

Работник – это лицо, расширяющее возможности пользователя. Данный тип роли подразумевает что у пользователя есть определенная должность и он работает в конкретном отделе компании.

Администратор – это пользователь который получает все возможности использования системы. Обычно в этой роли выступают директора и те, кому директора делегируют полномочия в системе.

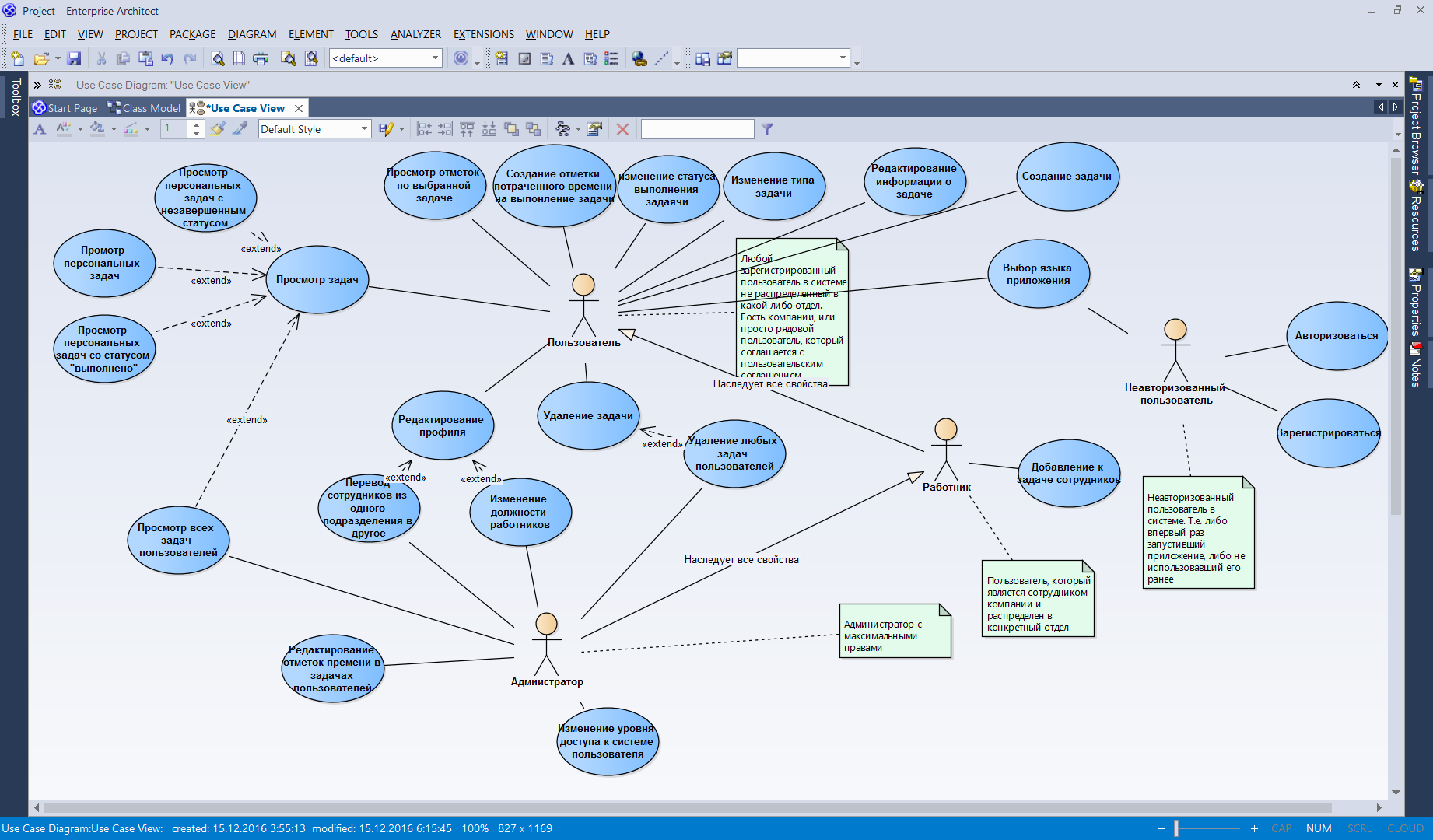


Рисунок 3.1 - Диаграмма вариантов использования системы

Для неавторизованного пользователя определены следующие функции использования системы:

* Авторизация;
* Регистрация;
* Выбор языка приложения.

Авторизация подразумевает под собой авторизацию в системе, чтобы в дальнейшем система могла идентифицировать личность пользователя и предоставлять функционал согласно регламентациям компании, на уровень доступа в систему.

Регистрация предполагает под собой возможность зарегистрироваться в системе, для использования функций которые доступны пользователю с его уровнем доступа.

Выбор языка приложения позволяет локализовать приложение под конкретный язык. При смене языка меняется язык интерфейса приложения, но не сами данные, которые пользователь вводит в системе.

Для пользователя системы определен следующий набор вариантов использования системы:

* выбор языка приложения;
* создание задачи;
* редактирование информации о задаче;
* изменение статуса выполнения задачи;
* изменение типа задачи;
* просмотр персональных задач, либо задач в которых участвует пользователь;
* просмотр выполненных персональных задач;
* просмотр выполненных задач, относящихся к пользователю;
* удаление задач, созданных пользователем;
* редактирование личного профиля;
* создание метки описывающей работу над задачей;
* просмотр отметок по выбранной задаче;
* удаление выбранной отметки по выбранной задаче.

Создание задачи предполагает под собой возможность создать новую задачу пользователем с наименованием, описанием выбором типа задачи указания даты начала и конца, а также указания процентной готовности.

Редактирование информации о задаче предполагает возможность изменения названия задачи и описания задачи. Задачу пользователь может редактировать только если он её создавал.

Изменение статуса выполнения задачи позволяет изменять статус задачи на «выполнено», или «в процессе». Что позволяет фильтровать задачи на «текущие» и «завершенные».

Изменение типа задачи позволяет изменить тип выбранной задачи.

Просмотр персональных задач, либо задач в которых участвует пользователь предоставляет возможность просматривать задачи, в которых пользователь может быть задействован, либо которые пользователь создал сам.

Просмотр выполненных персональных задач представляет собой отфильтрованные задачи пользователя по отметке «выполнено» созданных пользователем.

Просмотр выполненных задач, относящихся к пользователю представляет собой отфильтрованные задачи в которых участвует пользователь по отметке «выполнено».

Удаление задач, созданных пользователем даёт возможность удаления задач, созданных пользователем.

Редактирование личного профиля позволяет редактировать личную информацию пользователя, такую как фамилия, имя, отчество, почтовый ящик, мобильный телефон, домашний телефон, пол, дату рождения и город проживания или рождения.

Создание метки описывающей работу над задачей позволяет пользователю отмечать время работы над конкретной задачей, с названием действия и подробным описанием того, что делал пользователь в рамках задачи.

Просмотр отметок по выбранной задаче позволяет увидеть по выбранной задаче все отметки времени, чтобы узнавать куда тратилось время для выполнения задачи.

Удаление выбранной отметки по выбранной задаче позволяет удалить конкретную отметку времени по задаче, если по мнению пользователя она не лишняя или он допустил ошибку. Удалить пользователь может только свои добавленные отметки.

Работник наследует весь спектр вариантов использования системы пользователя, но также добавляется следующий функционал:

* добавление к задаче сотрудников.

Добавление к задаче сотрудников подразумевает под собой возможность добавлять других сотрудников компании и пользователей в системе к своей задаче, что поддерживает концепцию работы надо одной задачей нескольких людей. Для работы над одной задачей, пользователи могут быть в разных отделах.

Администратор получает все функции работника, но дополнительно обладает:

* удаление любых задач пользователей;
* изменение должности работников;
* перевод сотрудников из одного подразделения в другое;
* просмотр всех задач пользователей;
* редактирование отметок времени в задачах пользователей;
* изменение уровня доступа к системе пользователя.

Удаление любых задач пользователей позволяет администратору удалять любые задачи любых пользователей.

Администратор имеет право изменять должности работников, что сказывается на дальнейшей оплате труда работников.

Перевод сотрудников из одного подразделения в другое является одной из прерогатив администраторов, так как это позволяет переместить пользователя в другой отдел, что будет изменять статистику компании по подразделениям.

Администратор имеет возможность просмотреть и отредактировать все задачи пользователей, которые были созданы в системе.

Одна из главных возможностей администратора – это изменение уровня доступа к системе определенного пользователя. Свой уровень доступа изменять нельзя, если администратор хочет изменить свой уровень доступа к системе, то это подразумевает обращение к другому пользователю с правами администратора.

* 1. Выбор и обоснование архитектурных решений

На начальных этапах процесса проектирования должны быть приняты принципиальные решения, во многом определяющие этот процесс, а также качество и трудоемкость разработки.

На сегодняшний день очень большую популярность имеют программы, позволяющие превратить умный телефоне не только в центр развлечений, но и помощник для выполнения различных заданий и напоминаний. Мобильные приложения позволяют всегда иметь под рукой нужные компоненты для взаимодействия с различными источниками информации.

Основные преимущества мобильных приложений:

1. В отличие от мобильных сайтов, приложения очень быстро загружаются, так как они не «выкачивают» из сети картинки и прочие «тяжёлые» элементы;
2. Мобильные приложения могут работать в режиме оффлайн, то есть в то время, когда подключение к интернету отсутствует. Это огромное преимущество, так как оно позволяет компании всегда быть «на связи» со своим потенциальным клиентом, даже тогда, когда у него нет средств на счёте мобильного телефона. Как только пользователь сможет подключиться к сети, тогда все изменения будут автоматически загружены в его приложение.
3. Приложения создаются специально для конкретных моделей телефонов, поэтому они по-настоящему могут учитывать их специфику, предлагая пользователю максимально удобное управление. Все кнопки в приложении будут располагаться удобно и будут достаточно большими, благодаря учёту специфики аппарата.
4. Приложения удачно могут использоваться как маркетинговый инструмент. Пока ещё их рынок не слишком перенасыщен, поэтому сделать их популярными среди пользователей проще, чем, например, продвинуть в интернете простой мобильный сайт.
5. Факт запуска собственного приложения является сам по себе прекрасным рекламным поводом, который позволяет задействовать прочие инструменты маркетинга. К примеру, могут быть написаны новости на соответствующих ресурсах, могут быть разосланы пресс-релизы, даже телевизионная реклама может использовать событие в своих целях.
6. Запуск приложения направлен на очень широкую аудиторию, которая в последние годы весьма динамично растёт. Уже сейчас во всём мире продано около 2 миллиардов смартфонов, и их производство остаётся на высоком уровне, так как спрос на устройства постоянно увеличивается.
7. Приложения могут интегрироваться с навигаторами, которые чаще всего встроены в смартфоны. Это позволит в любое время на страничке контактов показывать путь до офиса компании от текущего местонахождения пользователя. Такие возможности не может давать никакой другой инструмент маркетинга и продвижения.
8. Мобильные приложения могут интегрироваться с сайтом, что позволит загружать на них все обновления, новости и информацию об актуальных акциях и скидках. Все клиенты будут в курсе изменений благодаря приложению на своём мобильном телефоне, при этом, не тратя время на загрузку непосредственно самой страницы в интернете.
9. Мобильные приложения позволяют настроить отслеживание электронной торговли и собирать сведения о всех приобретенных товарах, общем количестве покупок и их стоимости.
10. Мобильные приложения могут служить надежным каналом для безопасного обмена данными. Приложения позволяют пользователям быстро передавать, просматривать, загружать, редактировать и делиться друг с другом корпоративными данными прямо с мобильных устройств. При этом пользователи соблюдают корпоративные политики безопасности по работе с данными, что повышает их сохранность при удаленной работе.
11. Использование мобильного приложения формирует бренд компании в глазах клиентов, ведь в телефоне пользователя наряду с такими приложениями, как Youtube, Альфа банк, Google может быть приложение компании.

Для создания системы управления рабочим временем были выбраны использованы технологии:

1. android;
2. android sdk;
3. объектно-ориентированный язык программирования java;
4. библиотека отслеживания ошибок acra;
5. библиотека поддержки design;
6. библиотека упрощения работы с компоновкой элементов recyclevew;
7. android sqlite;
8. android studio.

Android - [операционная система](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) для [смартфонов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD), [интернет-планшетов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BF%D0%BB%D0%B0%D0%BD%D1%88%D0%B5%D1%82), [электронных книг](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BA%D0%BD%D0%B8%D0%B3%D0%B0_(%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE)), [цифровых проигрывателей](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D0%B8%D0%B3%D1%80%D1%8B%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C), [наручных часов](https://ru.wikipedia.org/wiki/%D0%A3%D0%BC%D0%BD%D1%8B%D0%B5_%D1%87%D0%B0%D1%81%D1%8B), [игровых приставок](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0), [нетбуков](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D1%82%D0%B1%D1%83%D0%BA), [смартбуков](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D0%B1%D1%83%D0%BA), [очков Google](https://ru.wikipedia.org/wiki/Google_Glass), [телевизоров](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D0%B2%D0%B8%D0%B7%D0%BE%D1%80) и других устройств. В будущем планируется поддержка [автомобилей](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C) и [бытовых роботов](https://ru.wikipedia.org/wiki/%D0%91%D1%8B%D1%82%D0%BE%D0%B2%D0%BE%D0%B9_%D1%80%D0%BE%D0%B1%D0%BE%D1%82). Основана на [ядре Linux](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_Linux) и собственной реализации [виртуальной машины Java](https://ru.wikipedia.org/wiki/Java_Virtual_Machine) от [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)). Данная операционная система была выбрана потому, что это самая популярная операционная система на мобильном рынке. на рисурнке 3.1 отображена диаграмма процентного присутствия на мобильном рынке операционных систем на июль 2016 года. Следует отметить плюсы мобильной операционной системы от компании Google:

* разнообразие приложений и игр;
* android – операционная система с открытым исходным кодом;
* в системе хорошо реализована многозадачность, когда без проблем работает одновременно несколько приложений;
* оперативные обновления, улучшающие функционал операционной системы, исправляются баги, вносятся изменения в интерфейс.

К минусам данной операционной системы можно отнести то, что старые версии систем были плохо оптимизированы под расход заряда аккумуляторной батареи. Так же бывают проблемы с совместимостью со устройствами снятыми с продажи или выпущенные безымянными фирмами. Обилие настроек для большинства пользователей представляют большую трудность.

Несмотря на все свои недостатки, операционная система Android была и будет востребована на рынке мобильных устройств. Основной причиной тому является лояльное отношение к пользователям.

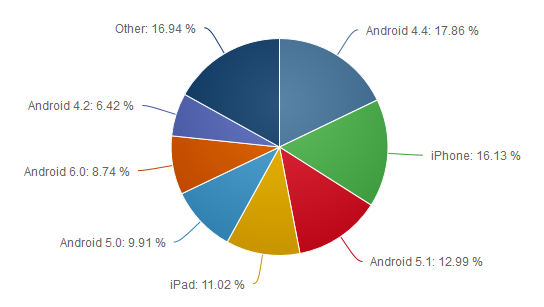


Рисунок 3.2 – процентное соотношение Android к другим опрационным системам

Android SDK - универсальная оболочка для моделирования и разработки различных программных продуктов под операционную систему Android. Здесь можно как создать, так и протестировать разработанные приложения с использованием достаточно широкого набора встроенных инструментов. Основанный на Linux, **Android SDK** использует виртуальное устройство для запуска приложений с поддержкой 3G, WiFi, GPS, сенсорных экранов, Bluetooth, компаса, акселерометра и других опций, которые являются сегодня неотъемлемой частью любого Android-устройства. Итогом использования этой среды станет качественно отлаженная и проверенная программа, готовая к публикации на Android Market, что также можно осуществить через **Android SDK.**

Кроме всего перечисленного, нужно отметить полную поддержку мультимедийного аудио- и видеоконтента самых разнообразных форматов, довольно качественное интегрирование с браузерами, работу с базами данных SQLite и многие другие полезные и удобные инструменты для помощи в разработке.

Объектно-ориентированный язык программирования Java - язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения. Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) — программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байт-код, в отличие от текста, обрабатывается значительно быстрее. Язык Java зародился как часть проекта создания передового программного обеспечения для различных бытовых приборов. Реализация проекта была начата на языке [C++](http://progopedia.ru/language/c-plus-plus/), но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента — языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами. Для приложений написанных под операционную систему Android происходит перекомпиляция кода на языке Java в байт-код dalvikVM, по причине того, что Android устройства в основном работают на архитектуре ARM и обслуживание JVM крайне сложная операция. Подробно процесс построения и сборки проекта представлен на рисунке 3.2.

**Android** предоставляет несколько способов для хранения пользовательских данных и приложений. **SQLite** является одним из способов хранения данных пользователя. SQLite это очень легковестная база данных, которая содержится в **ОС Android предоставляется API оболочки с совместимым интерфейсом. API Android SQLite является типичным; разработчику следует реализовать всю обработку базы данных, включая создание, управление версиями, обновления базы данных и прочие настройки. Если нужно использовать заранее заполненную базу данных SQLite, требуется дополнительная настройка. К достоинствам этой базы можно отнести то, что эта база встроена в саму систему и разработчику не надо задумываться об установке системы управления базами данных в телефон. Так же эта SQLite имеет поддержку внешних ключей, что позволяет ей поддерживать целостность базы данных. К недостаткам данной базы данных можно отнести то, что она поддерживает мало типов данных.**

**Android studio – это** [**интегрированная среда разработки**](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) **(IDE) для работы с платформой** [**Android**](https://ru.wikipedia.org/wiki/Android)**, анонсированная** [**16 мая**](https://ru.wikipedia.org/wiki/16_%D0%BC%D0%B0%D1%8F)[**2013**](https://ru.wikipedia.org/wiki/2013) **года на конференции**[**Google I/O**](https://ru.wikipedia.org/wiki/Google_I/O)**. Новые функции появляются с каждой новой версией Android Studio. На данный момент доступны следующие функции [5]:**

* **расширенный редактор макетов:** [**wysiwyg**](https://ru.wikipedia.org/wiki/WYSIWYG)**, способность работать с** [**ui**](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) **компонентами при помощи** [**drag-and-drop**](https://ru.wikipedia.org/wiki/Drag-and-drop)**, функция предпросмотра макета на нескольких конфигурациях экрана.**
* **сборка приложений, основанная на** [**gradle**](https://ru.wikipedia.org/wiki/Gradle)**.**
* **различные виды сборок и генерация нескольких** [**.apk**](https://ru.wikipedia.org/wiki/.APK) **файлов**
* [**рефакторинг**](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3) **кода**
* **статический анализатор кода (lint), позволяющий находить проблемы производительности, несовместимости версий и другое.**
* **встроенный proguard и утилита для подписывания приложений.**
* **шаблоны основных макетов и компонентов android.**
* **поддержка разработки приложений для** [**android wear**](https://ru.wikipedia.org/wiki/Android_Wear) **и** [**android tv**](https://ru.wikipedia.org/w/index.php?title=Android_TV&action=edit&redlink=1)
* встроенная поддержка google cloud platform, которая включает в себя интеграцию с сервисами google cloud messaging и app engine.

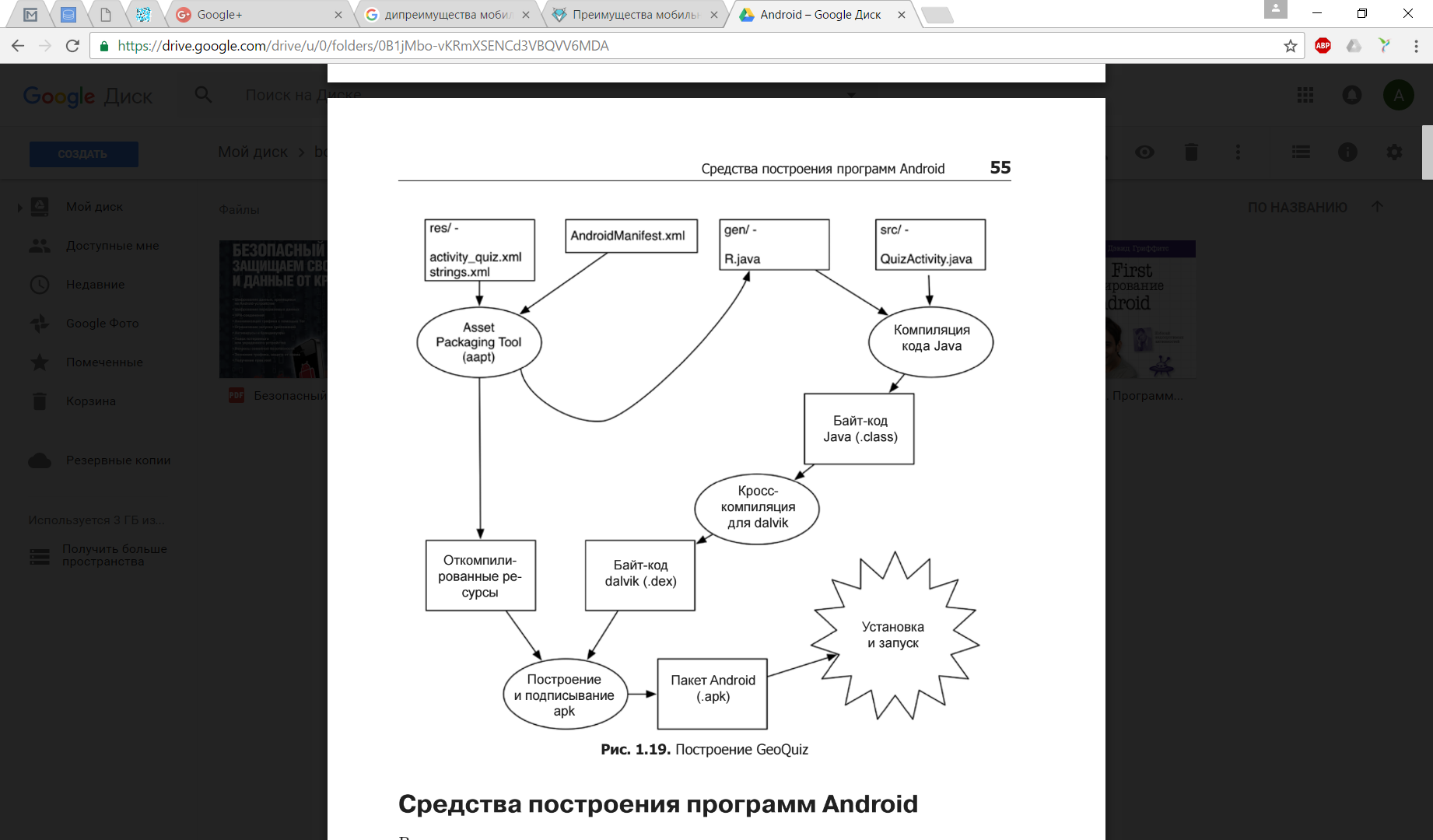


Рисунок 3.2 – Этапы построения Android приложения

Библиотека отслеживания ошибок ACRA –это бесплатная библиотека для создания и отправки отчетов об ошибках в Android приложениях. Acra фиксирует сбои и ошибки и может отправлять данные на специально заданный сервис или в google-таблицы. При этом в данных передаются различные параметры устройства и другие технические значения. Основное отличие ACRA от Crashlytics в том, что продукт является открытым и распространяется свободно. Также разработчики могут самостоятельно выбирать куда отправлять данные. Например, разработчики могут разместить сервер сбора данных AcraLyzer на своем сервере и анализировать аналитику без предоставления данны внешним сервисам. Из достоинств данной библиотеки можно отметить что она отправляет полную информацию об исключениях, включая стек вызовов, модель устройства, информацию о системе и другие настраиваемые параметры. К минусам же можно отнести то, что Раньше ее можно было сконфигурировать для работы с Googe Drive, когда отчеты об ошибках шли в документ. Сейчас, когда Googe прикрыл это дело, приходится настраивать свой сервер на основе CouchDB, что само по себе не просто. Или использовать существующий хостинг для CouchDB.

Библиотека поддержки Design - Новая библиотека поддержки **Android Design Support Library** делает доступным весь набор компонентов материального дизайна для всех версий, начиная с **Android 2.1** и выше:

* navigation view (для navigation drawer) — панель навигации;
* floating labels (для edittext) — плавающий ярлык;
* floating action button (fab) плавающая кнопка;
* snackbar — всплывающее уведомление с кнопкой;
* tabs — вкладки;
* motion and scroll framework — управление жестами и прокруткой.

Библиотека упрощения работы с компоновкой элементов RecycleVew – это замена старым компонентам системы таким как ListView, который позволяет более экономно и разумно использовать память. Данная библиотека была представлена компанией Google на ежегодной конференции для разработчиков IO 2015/ Главным отличием RecyclerView является создания всех элементов списка единожды, в отличии от ListView, где каждая View в списке создавалась отдельно для каждого из элементов списка, что в свою очередь приводило к огромному использованию памяти при создании достаточно больших списков. Так же данная библиотека обязует использовать паттерн ViewHolder пример этого паттерна приведен на рисунке 3.3. Данный паттерн обязует реализовать отдельный класс для прорисовки компонентов, что при правильном использовании компонентов позволит уменьшить затраты на выделение памяти. Класс отвечающий за отображение связывается с классом адаптера и создает прорисовывающих объектов только, сколько может поместиться на экране, и не более того.

Большим преимуществом RecyclerView является то, что в нем можно задавать различные классы, которые отвечают за компоновку элементов на экране. Например [LinearLayoutManager](https://developer.android.com/reference/android/support/v7/widget/LinearLayoutManager.html) располагает компоненты так, чтобы они отображались последовательно в зависимости от ориентации компоновки.

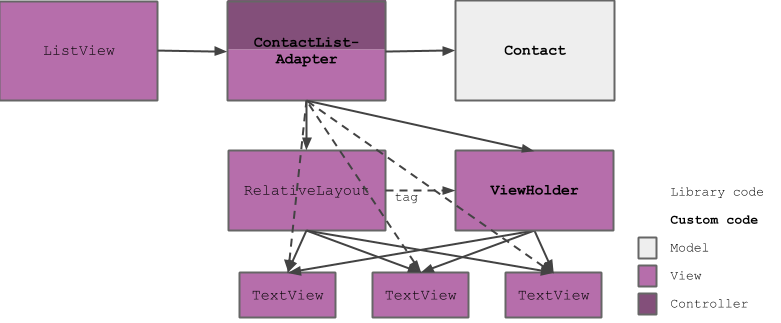


Рисунок 3.3 – Паттерн ViewHolder

[GridLayoutManager](https://developer.android.com/reference/android/support/v7/widget/GridLayoutManager.html) – компонент отвечающий за компоновку элементов в виджете в виде сетки, размерность которой указывает разработчик.

* 1. Функциональное моделирование процесса управления рабочим временем

Теперь рассмотрим сам процесс автоматизации в нотации IDEF0. На рисунке А.1 представлена контекстная диаграмма данного процесса.

Как видно из контекстной диаграммы, в разработанном проекте проходила автоматизация процесса планирования рабочего времени. Данный процесс был значительно изменён с внесением в его состав методики и технологии тайм-менеджмента. Входящими данными данного процесса является рабочее время, которое предоставляется сотрудникам, а на выходе получаем спланированное рабочее время сотрудников. Основными механизмами, выполняющими данный процесс, являются информационная система, на которой установлено программное обеспечение, финансовые и информационные ресурсы, а также оборудование, на котором можно будет запустить информационную систему. В качестве управляющих элементов представлены номенклатура, задания пользователей, а также документация по их составлению.

Рассмотрим первый уровень декомпозиции данного процесса. Данная декомпозиция представлена на рисунке А.2.

Как видно из рисунка, процесс планирования времени будет теперь представлен тремя подпроцессами: подготовка к работе, выполнение работы, анализ результатов выполненной работы. Для начала сотруднику необходимо подготовиться к выполнению работы. Для того, чтобы наиболее производительно выполнить работу с минимальными затратами, необходимо, чтобы в подготовке к работе пользователь пользовался советами или методиками тайм-менеджмента. Поэтому при декомпозиции данного процесса, представленного на рисунке А.3, видно, что в начале сотрудник должен определить цель работы, причём сформулировать её корректно и по технологии SMART, после чего со сформировавшейся целью сотрудник должен выбрать методику для управления временем, после чего подобрать и подготовить инструменты для определённой методики. После того, как данные процессы будут выполнены, сотрудник будет располагать оставшимся для работы временем. Если рассмотреть механизмы, влияющие на данный процесс, то тут можно отметить участи информационной системы, в которой будет происходить планирование и управление временем, а также финансовый и информационные ресурсы, которые используются в данных процессах.

Рассматривая процесс выполнения работы можно отметить, что при правильном выборе методики управления временем данный процесс может включать в себя различные подпроцессы по контролю деятельности.

Если рассматривать процесс анализа результатов, то тут можно отметить следующие подпроцессы: просмотр данных по результату проделанной работы, анализ результатов, план управления временем на следующий период исходя из полученных данных по выполнению работы. Следует отметить, что при просмотре данных о результатах работы пользователь будет заходить в систему, где будет представлена статистика по выполнению задач пользователем. Анализ результатов в основном проходит при непосредственном изучении статистики, оценке прибыли от выполненной работы, получения оценки от непосредственного руководителя или сотрудника, который делегировал данную задачу, а также формирование отчёта о выполненной, где будут представлены данные по продолжительности выполнения работы, количества задач, которые присутствовали в работе и на которые было разбито общее задание, а также данные по эффективности управления временем в виде графиков и диаграмм. Декомпозиция процесса анализа результатов представлена на рисунке А.4.

Данные, которые будут получены по окончанию процесса управления временем сотрудника, помогут в дальнейшем выбирать методики, определять оптимальное количество задач, которые необходимы для выполнения, правильно определять их продолжительность, а также определять периоды времени, когда работа сотрудника будет наиболее продуктивна. Также данная информация будет полезна для руководителя, так как предоставление отчёта о проделанной работе будет описывать производительность сотрудника, насколько быстро и хорошо он выполняет свою работу и можно ли его загружать больше или стоит снизить нагрузку. Данные, которые представлены в виде графиков, отлично подходят для анализа, поэтому просто определить статистику по данным, прогресс, которого добился сотрудник при применении определённой методики, ведь все они довольно индивидуальны. В общем, анализ полученных результатов представляет большую роль для сотрудника в общей системе тайм-менеджмента, и автоматизация данного процесса ещё больше сократит расходы времени, так как не нужно это делать вручную.

Таким образом, система должна быть спроектирована таким образом, чтобы все вышеуказанные задачи выполнялись, а эффективность деятельности сотрудника максимизировалась. Данные, которые будут поступать на вход системы, должны быть представлены в виде данных о рабочем времени сотрудника, причём на выходе, особенно при многократных повторениях процесса управления временем, будет создан наиболее оптимальный план работы сотрудника, что будет являться достижением поставленной цели. Также следует отметить, что процесс управления не должен быть простой последовательностью действий, а являться циклом, который будет принимать и предоставлять информацию в течении определённого времени. Далее будет рассмотрена система в целом. Это предполагает рассмотрение процессов разработки, проектирования, описания системы как она есть и других процессов.

* 1. Информационное моделирование системы управления рабочим временем

Информационные модели — это модели, описывающие классы объектов, принадлежащие к данной системе, атрибуты этих сущностей и отношения между этими сущностями. Построение информационной модели является сложным и многоэтапным процессом, но в последствии при построении правильной информационной модели система может экономить огромное количество ресурсов.

Для создания информационной модели использовался механизм программного обеспечения, предоставляющий графический, интуитивно понятный интерфейс MySQL Workbench который позволяет создавать таблицы на основе продуманных сущностей и не требует углубленных и профессиональных знаний по SQL для построения базы данных.

Данная информационная модель изображенная на рисунке отображает бизнес-логику корпоративной системы управления временем.

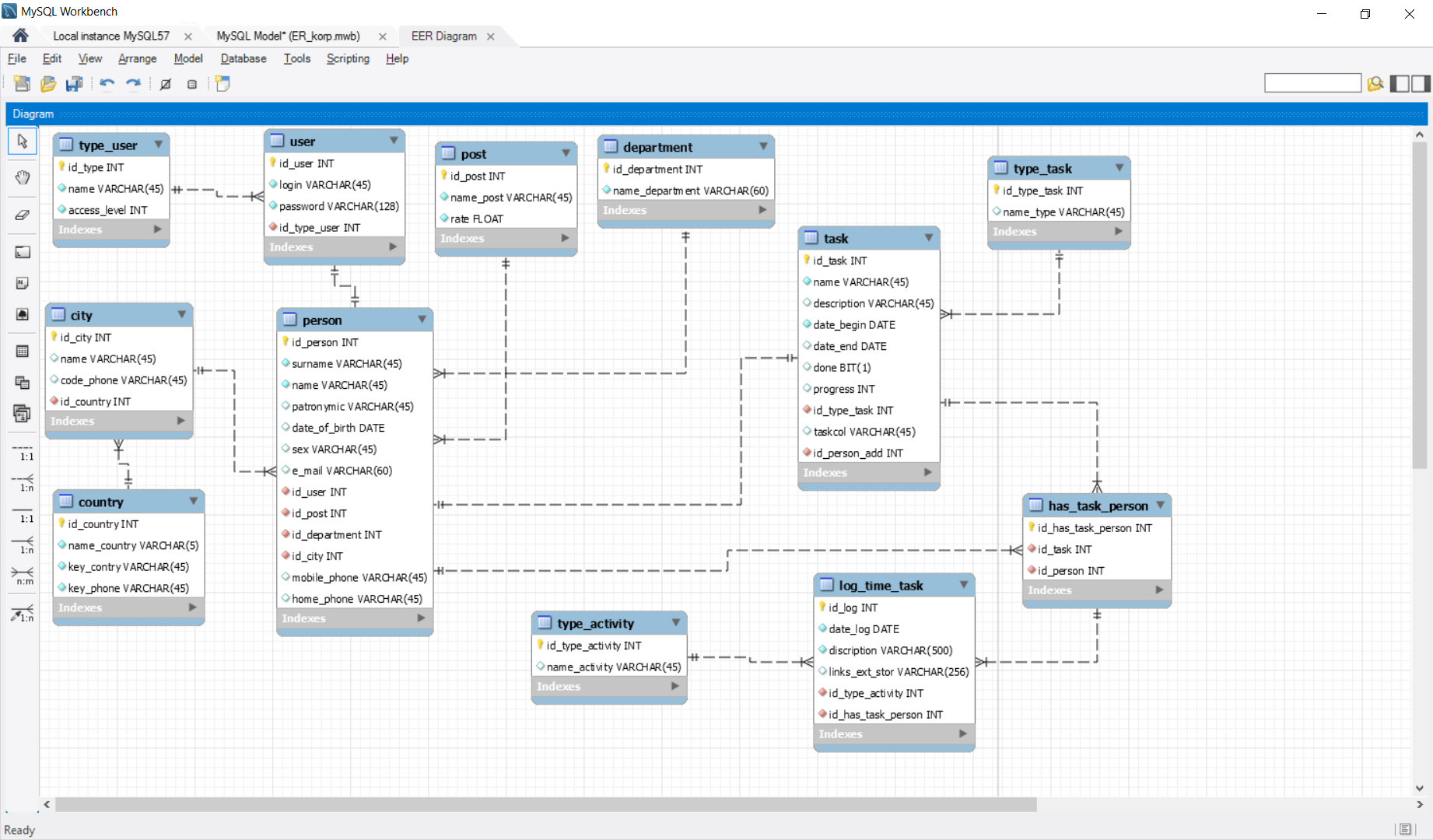


Рисунок 3.4 – Информационная модель системы управления временем сотрудников ИТ-компании

Одной из ключевых таблиц для всей системы является таблица user, в которой описывается сущность пользователя системы, его логин или электронная почта, пароль для входа в систему в зашифрованном виде по алгоритму SHA-1 ссылку на таблицу типа пользователей и идентификатор в таблице.

Таблица person хранит в себе такие поля как surname которое хранит в себе данные о фамилии. Поле name хранит в себе данные об имени пользователя. Поле patronymic хранит в себе данные об отчестве. Поле date\_of\_bith хранит информацию о дате рождения. Поле sex хранит информацию о половой ориентации пользователя. Поле e-mail хранит почту пользователя. Данная таблица связанна с рядом других таблиц некоторые из которых тоже связанные с другими таблицами. Данная таблица связанна с такими таблицами, как: department, past, city, task\_has\_person, поговорим о них подробнее.

Таблица department хранит в себе информацию об отделе, в котором работает данный сотрудник. Поле name\_department хранит в себе наименование отдела, в котором работает сотрудник.

Таблица post данная таблица указывает какую должность занимает сотрудник. Поле name\_post хранит в себе информацию о названии должности которую занимает сотрудник. Поле rate показывает какого графика должен придерживаться данный сотрудник.

Таблица city показывает город в котором живёт данный пользователь. Поле name хранит в себе название этого города. Так же таблица содержит поле key\_phone хранящее в себе код города. Данная таблица соединена с ещё одной таблицей country.

Таблица country хранит в себе информацию в какой стране живет данный пользователь и основную информацию о стране. Поле country\_name хранит в себе название страны. Поле key\_country\_name хранит в себе краткое название страны в которой находится сотрудник. Поле key\_phone хранит в себе начальные значения телефонных номеров, которые относятся к той или иной стране.

Таблица has\_task\_ person хранит информацию о задании, которое должен выполнить сотрудник. Она соединена с таблицей task и log\_time\_task.

Таблица task. Поле name хранит в себе название задания, полученного сотрудником. Поле data\_begin хранит в себе информацию о начале даты выполнения. Поле data\_end хранит в себе информацию об окончании работы над проектом сотрудником компании. Поле type\_task хранит в себе информацию о типе самого задания, которое было поручено сотруднику. Поле done хранит в себе информацию о статусе задания, выполняется или не выполняется. В таблице присутствует поле progress, которое указывает на процентную завершенность проекта. Имеется поле id\_add\_person, которое указывает на создателя задачи. Сделано это для того, чтобы можно было определять кто создает задачи и кто ответственен за задачу.

Таблица log\_time\_task хранит в себе какую-то контрольную точку, или временную отметку по проекту и содержит в себе поле date\_log, которое указывает на дату отметки какой-то активности по проекту. Description содержит описание временной отметки по задаче. Так же присутствует поле id\_has\_task\_person ссылающееся на запись в таблице has\_task\_person, то есть каждый участник в задаче оставляя лог, в базе оставляет след отметки времени по задаче. В таблице присутствует поле id\_type\_activity, в котором хранится ссылка на значение с таблицы type\_activity.

Таблица type\_activity содержит наименование активности, один вариант которого пользователь выбирает из существующих в базе при отметке времени активности по задаче.

После построения связей между базами данных требуется проверить, можно ли их использовать при создании проекта. Чтобы была возможность использовать данную систему связей таблиц, нужно проверить является ли данная система годной для поставленной задачи. Есть определённый свод правил, которым надо следовать при создании такого рода систем баз данных. Во-первых, система баз данных должна относится к нормальным системам баз данных третьего рода. Что в себя включает нормальная система баз данных третьего рода. Во-вторых, она не должна иметь повторяющихся колонок. В-третьих, должна иметься таблица с первичными ключами и должны иметься другие таблицы, зависимые от данных кочевых значений.

Данная модель подходит для создания проекта и соответствует всем нормам нормальности, данное умозаключение было принято после осмотра и проверки всех пунктов, которые требуются для соблюдения при создании нормальной модели базы данных.

* 1. Разработка моделей представления системы управления рабочим временем

Приведем описание диаграммы последовательности, представленной на рисунке 3.5. Данная диаграмма описывает один из процессов вывода информации на экран пользователя. При отображении используется много асинхронных процессов для более быстрой отзывчивости системы. Так же асинхронная загрузка данных должна использоваться в приложениях Android, чтобы не загружать основной поток, отвечающий за отображение макета и информации. Слишком долгое ожидание в главном потоке ответа от какой-либо функции вызывает ошибку ожидания, которая может негативно повлиять на работу приложения в целом и сложить негативное впечатление от программного продукта у конечного потребителя. Так же на этой диаграмме показан механизм работы RecyclerView и ViewHolder-механизма. Когда фрагмент в приложении получает набор данных от LoaderTaskData, то он создаёт объекты представлений необходимые для отображения информации. Пользователь по запросу вывода задач, получает от приложения отклик и на экране отображается прогресс загрузки данных, после плавно и динамично происходит процесс отображения данных в виде прокручиваемого списка, что позволяет смотреть задачи просто и без каких-либо задержек, а при большом количестве набора результирующих данных, ещё и сэкономить оперативную память.

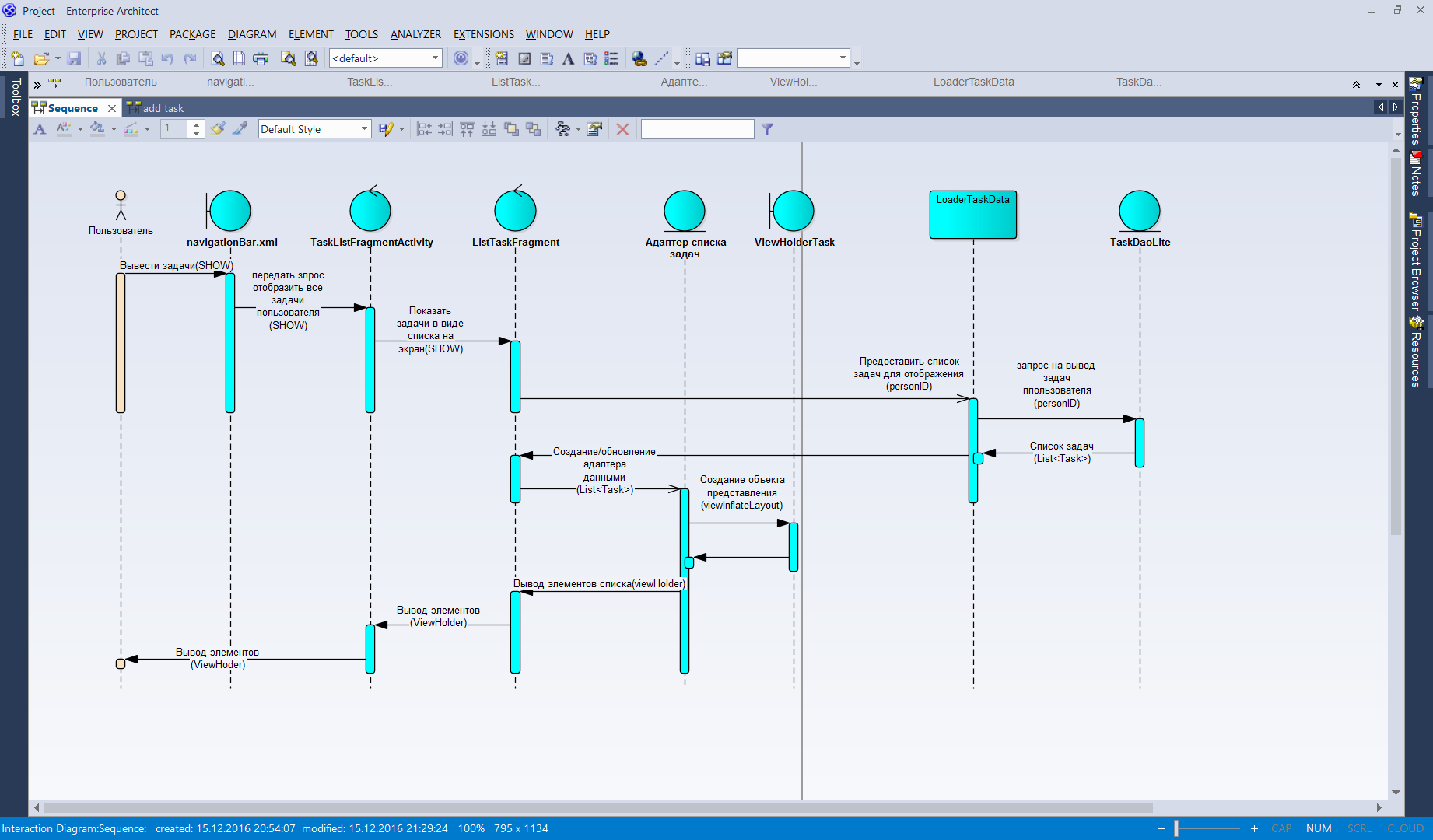


Рисунок 3.5 – Диаграмма последовательности вывода списка задач на экран

Диаграмма состояний приведена на рисунке 3.6. Как показано на диаграмме, то если пользователь не зарегистрирован в системе, то первоначальная активность переведет программу в состояние авторизации, где пользователь должен будет либо ввести данные необходимые для авторизации, либо перевести программу в состояние регистрации. При выполнении корректно одной из операций, пользователь переходит в личный кабинет, где может перевести в одно из других состояний (на диаграмме отображены).

На рисунках 3.7 и 3.8 представлены диаграммы классов, размещенные в разных пакетах и отмечающие за разные уровни взаимодействия в приложении.

В пакете controllers находится классы, которые принимает запросы, обрабатывает пользовательский ввод, взаимодействует с моделью и представлением и возвращает пользователю результат обработки запроса.

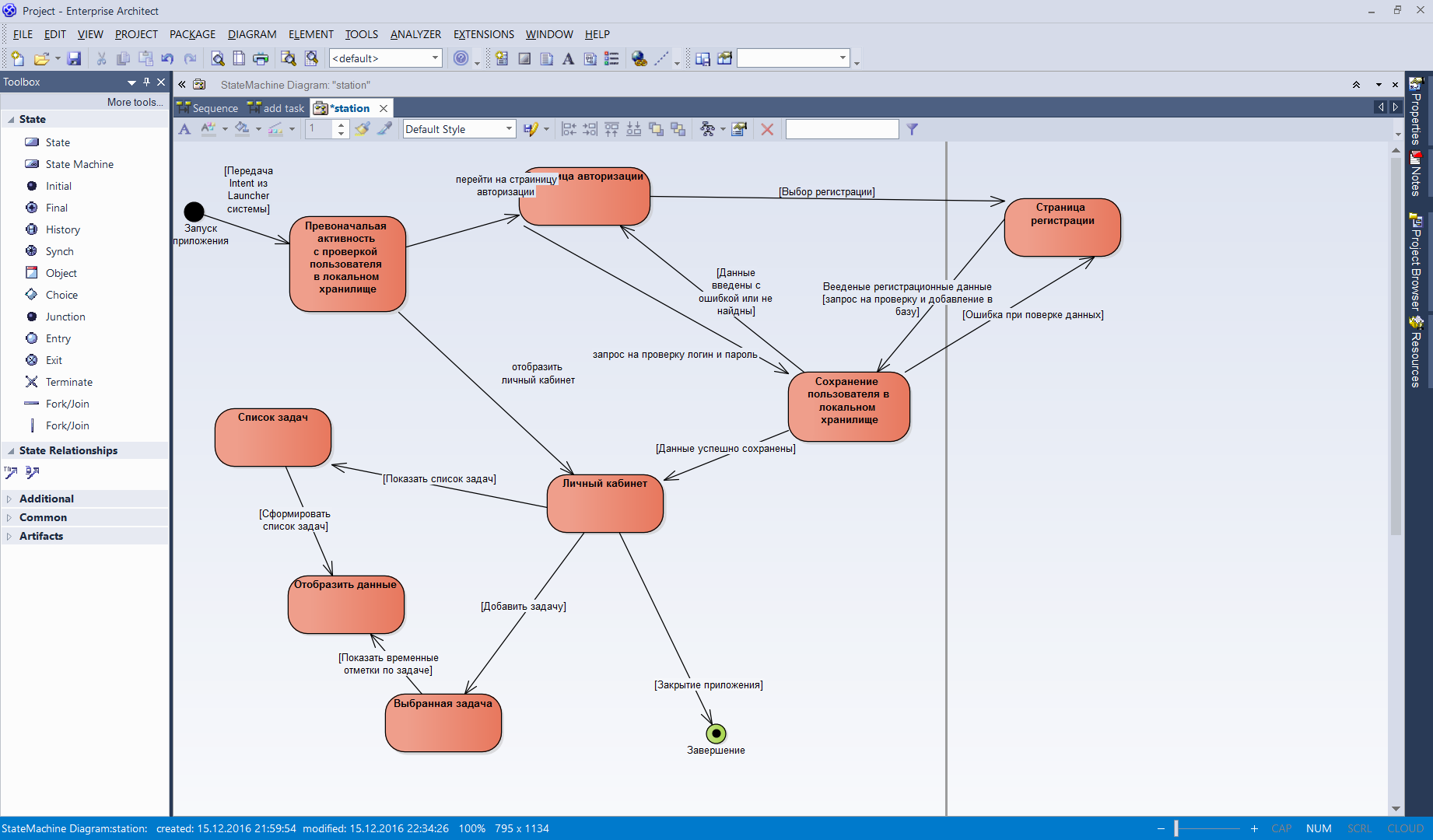


Рисунок 3.6 - Диаграмма состояний

На диаграмме классов пакета contollers отображена иерархия классов, которые выполняют роль контроллеров и связывают макеты, описанные в xml файлах с программными интерфейсами. Перед использованием макета в приложении базовые классы, от которых наследуется разработчик, преобразуют xml файлы в структурированный связующий граф, в котором вершины представлены как виджеты пользовательского интерфейса.

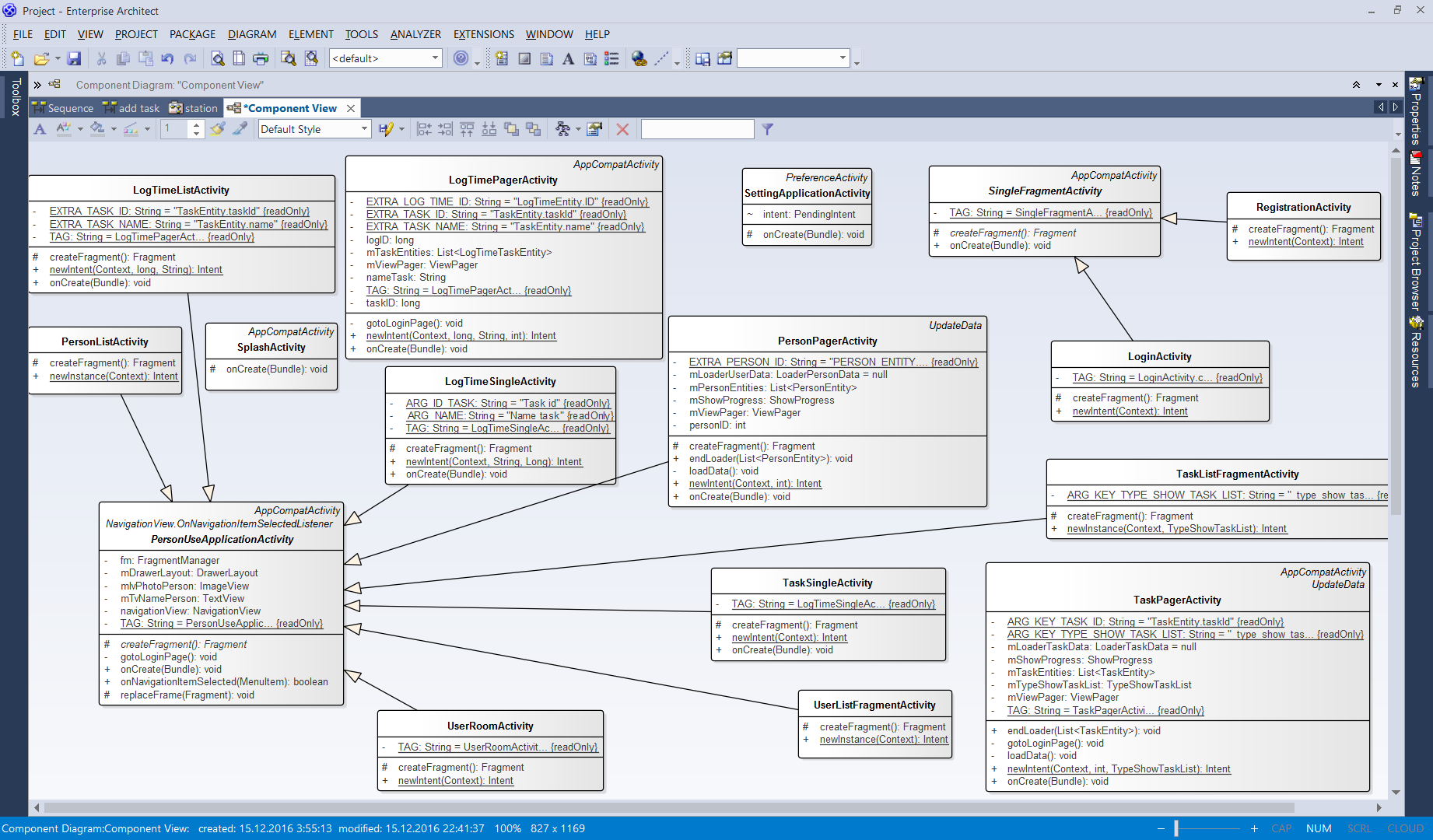


Рисунок 3.7- Диаграмма классов пакета controllers

На рисунке 3.8 В пакете model находятся классы, которые соответствуют сущностям базы данных и представляют слой, описывающий логику организации данных в приложении, а также классы, которые отвечают за обработку данных полученных с базы данных.

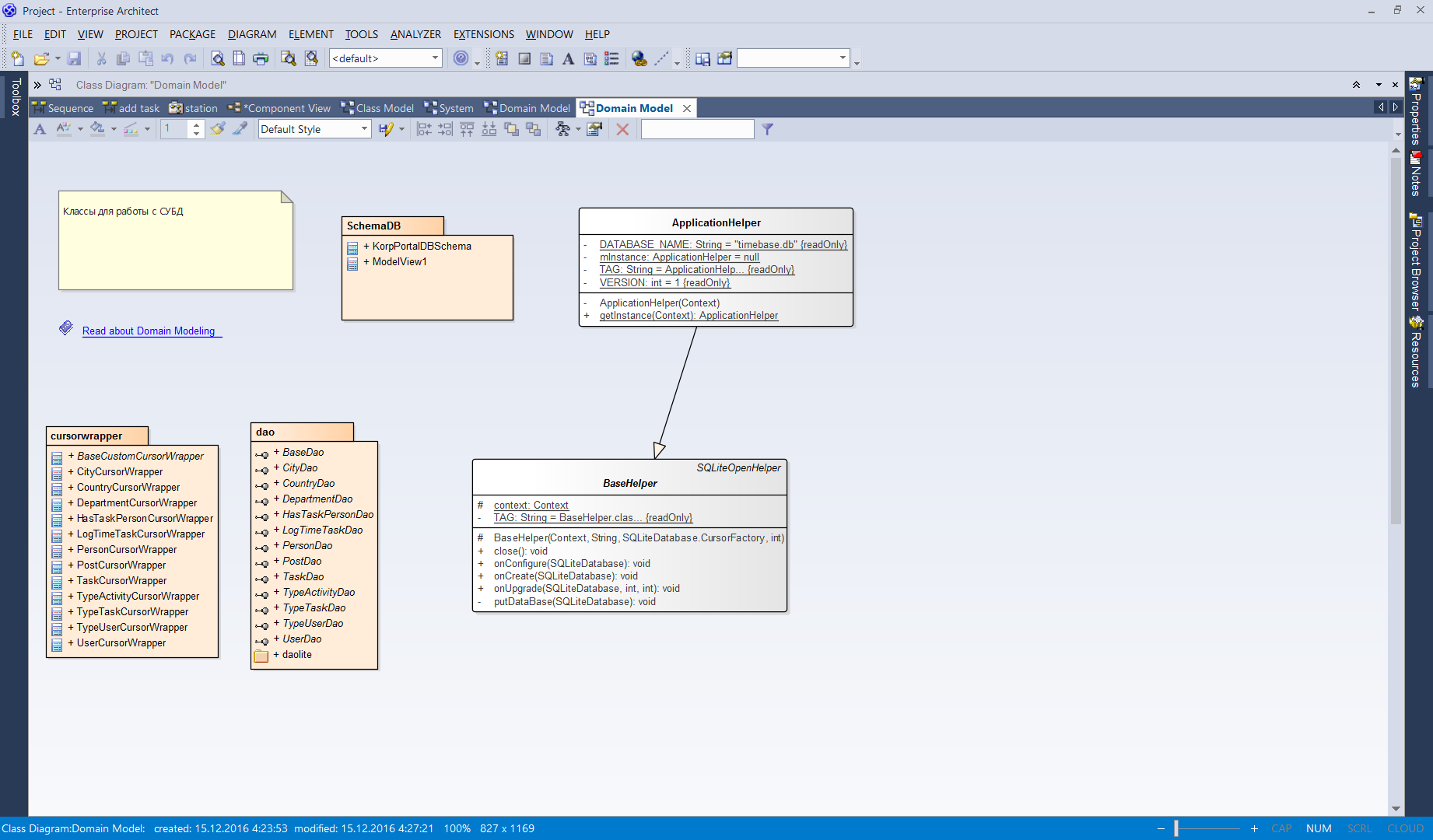


Рисунок 3.8 – Диаграмма классов пакета model

В диаграмме компонентов описаны элементы, без которых наша программа не может работать. Главным конфигурационным файлом приложения является AndroidManifest в котором прописана активность по умолчанию, все активности приложения, службы, разрешения для приложения использования системных ресурсов. Так же в данном приложении не обойтись без зависимости библиотек поддержки, которые позволяют оптимизировать интерфейс пользователя. Так же приложению не обойтись без встроенных системных библиотек, например, для асинхронной выгрузки данных, или отправки интентов в систему. Так же в компоненты приложения входят компилированные классы, файлы ресурсов, которые индексируются по определенному правилу, и из приложения к ним можно обратиться по уникальному идентификатору.

Диаграмма компонентов представлена на рисунке 3.9. Как видно на диаграмме, наше приложение включает в себя ряд библиотек, файловых ресурсов и макетов отображения.

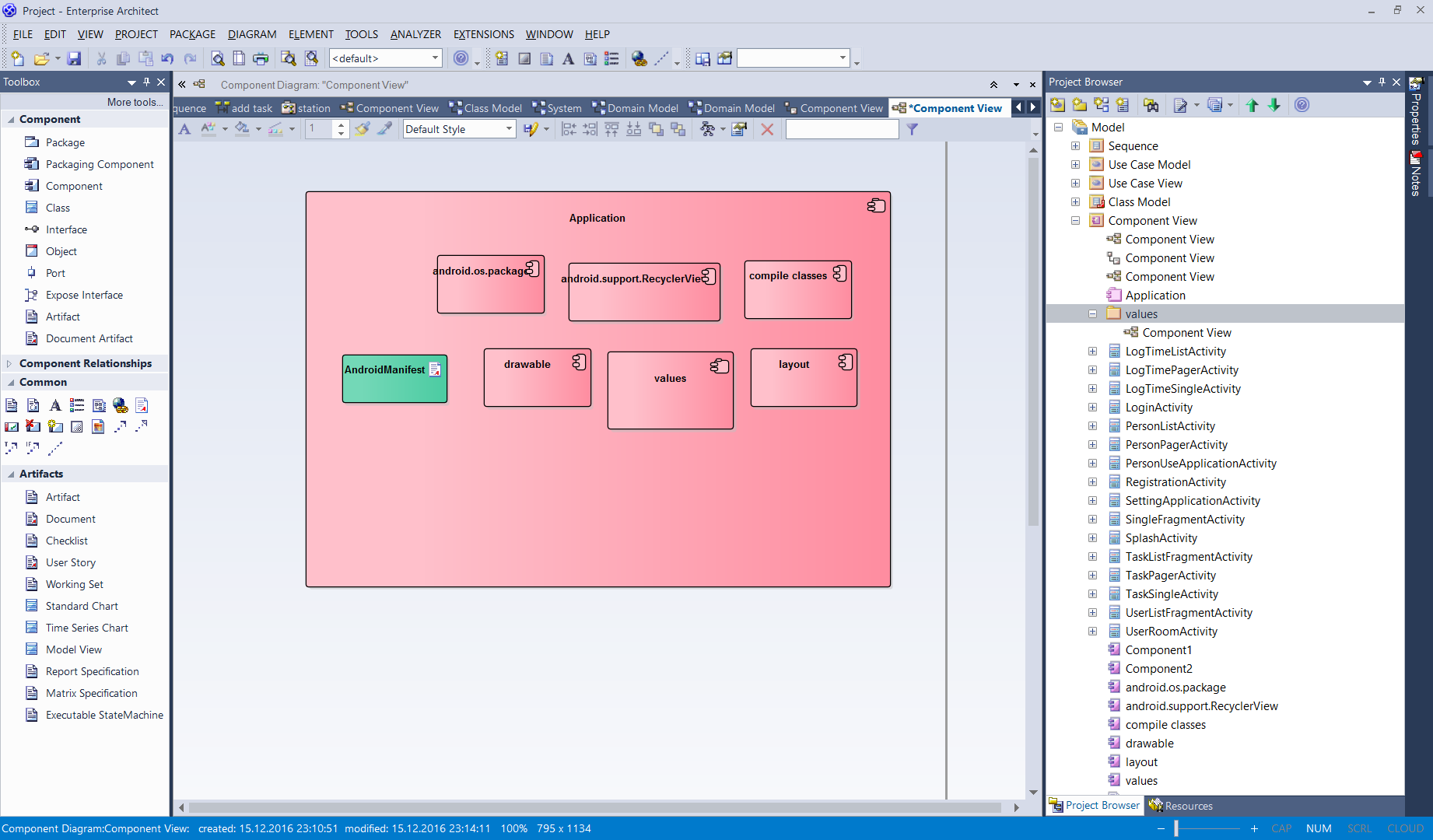


Рисунок 3.9 – Диаграмма компонентов

На рисунке 3.10 изображена диаграмма развёртывания. Так как это проект локальный, то все компоненты строены в ядро системы.

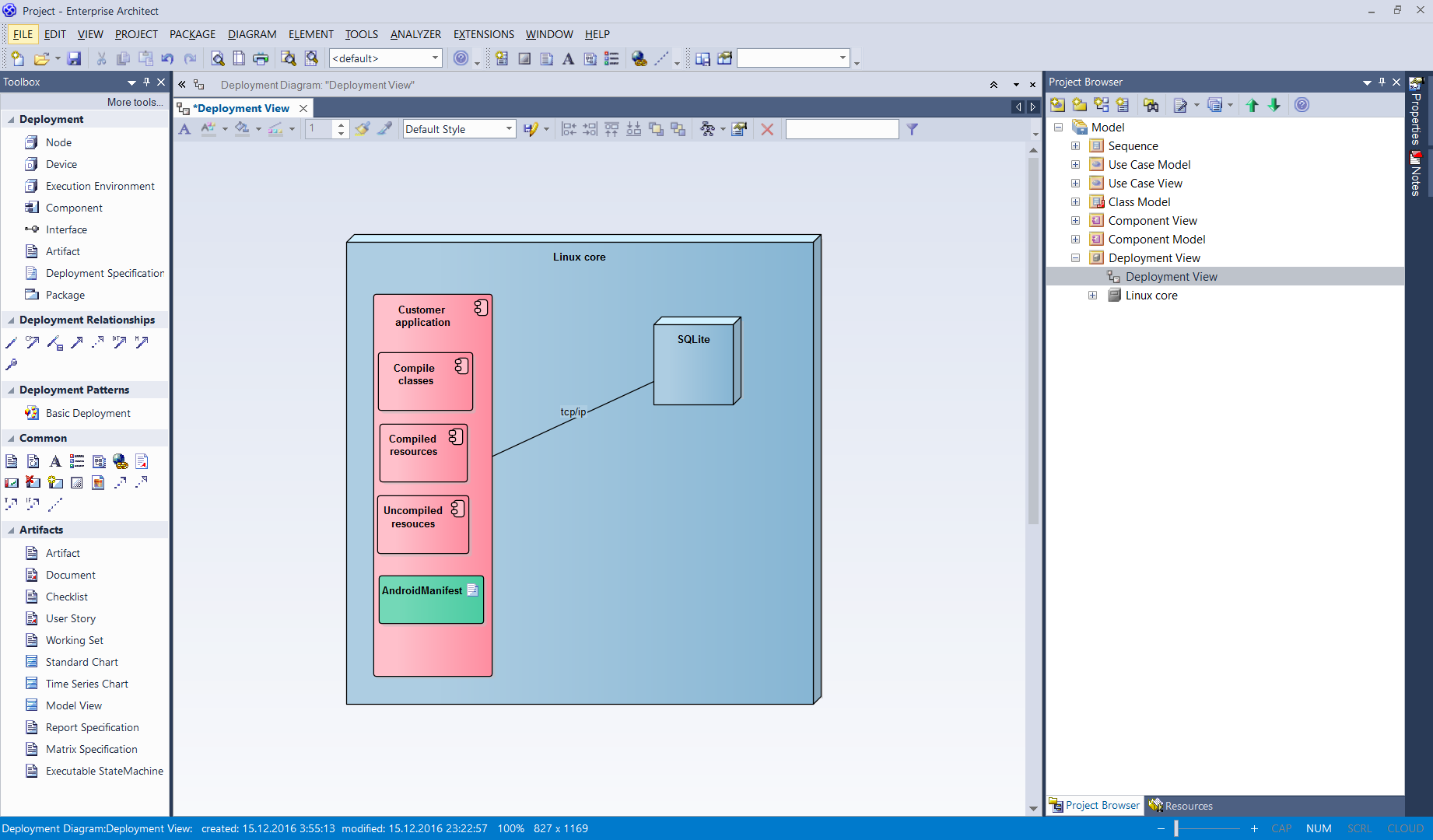


Рисунок 3.10 – Диаграмма развертывания

В результате было получено достаточно четкое представление о том, какому перечню требований должно соответствовать разрабатываемое приложение.

* 1. Описание алгоритмов программных модулей системы управления рабочим временем

Схема алгоритма, представленная на рисунке Б.1, в приложении Б показывает процесс добавления отметки времени по определенной задаче в случае, когда пользователь авторизовался в системе, и может работать с задачами.

После успешного входа в систему, пользователь может выбрать список задач, если задач у пользователя нет, то будет предложено добавить новую задачу.

Если задача существует в системе, то есть возможность по выбранной задаче добавлять отметки времени активности над выбранной задачей. Но тут есть ограничивающие условия, если пользователь не добавлен в выбранной задаче, то добавлять лог он не может, что будет выведено на экран в виде сообщения.

* 1. Результаты проектирования системы управления рабочим временем

В результате проектирования, разработки и переноса диаграмм и схем на код, получилось мобильное приложение с собственным дизайном, локально работающее на смартфоне, поддерживающее 3 рабочих роли в системе. Рассмотрим основные функции приложения по управлению рабочим временем в ИТ-компании.

Основных роли в приложении, как было раньше отмечено, выделено 3. У каждого своего права на работу в системе.

При запуске приложения перед пользователем отображается главный экран авторизации приложения, показанный на рисунке 3.11.

Войдём в систему под пользователем Андриан. Для этого нужно ввести определённые логин и пароль и перейдём к экрану главного экрана приложения, авторизованного пользователя, отображённого на рисунке 3.12. На экране авторизации доступна возможность регистрации новых пользователей. Это сделано для того, чтобы любой человек мог зарегистрироваться и пользоваться системой в личных целях.

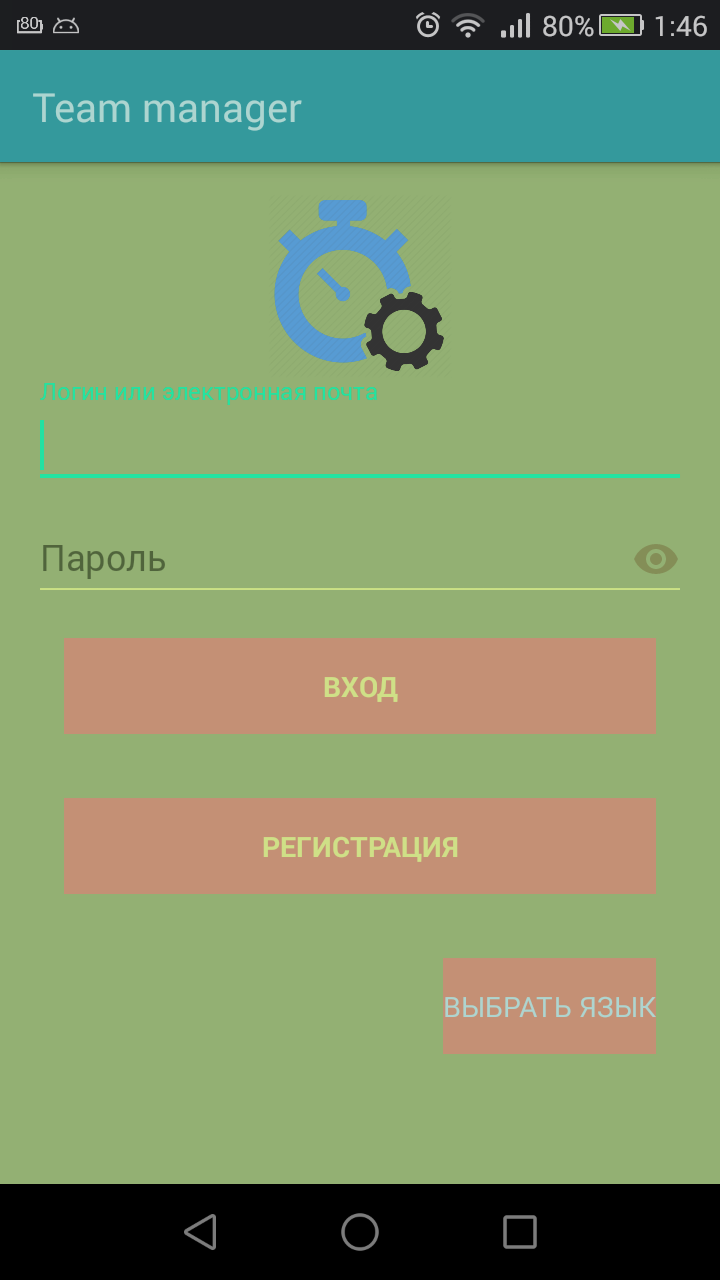


Рисунок 3.11 – Экран авторизации

В экране личного кабинета возможно редактировать личную информацию, путём нажатия на одно из доступных полей. Система сфокусируется на этом поле и появится клавиатура, на которой можно вводить редактируемую информацию. Создадим пользователю задачу, для этого необходимо провести пальцем от левого края экрана и выбрать пункт «создать задачу», после чего система перейдёт к другому экрану, где будет предложено ввести информацию о задаче и возможность добавления её в систему.

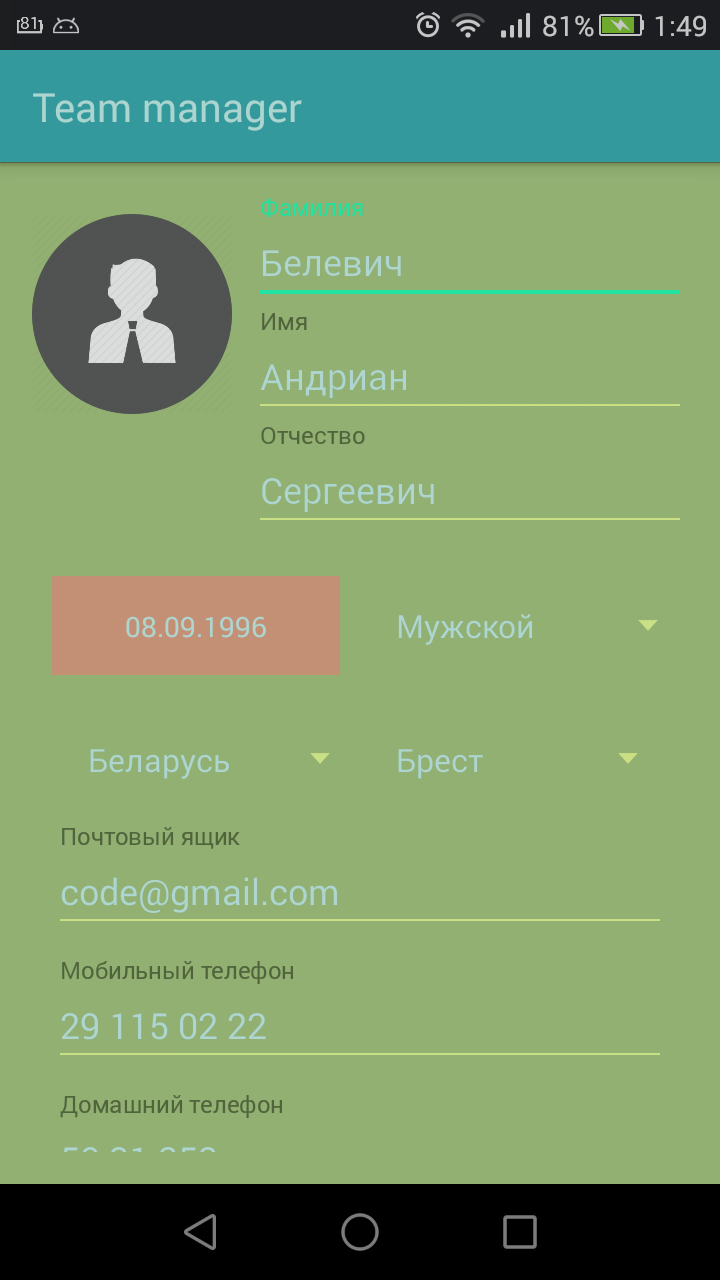


Рисунок 3.12 – Экран личного кабинета

Выезжающая панель, изображённая на рисунке 3.13 является одним из компонентов поддержки для старых версий операционных систем, и позволяет создавать красивые и одновременно функциональные приложения с облегченным доступом к многим компонентам системы. В зависимости от типа пользователя, в этом меню отображается разные функционал. Такая возможность была достигнута, путем вынесения пунктов для каждого типа пользователя в отдельные файлы, что придало дополнительную гибкость приложению.

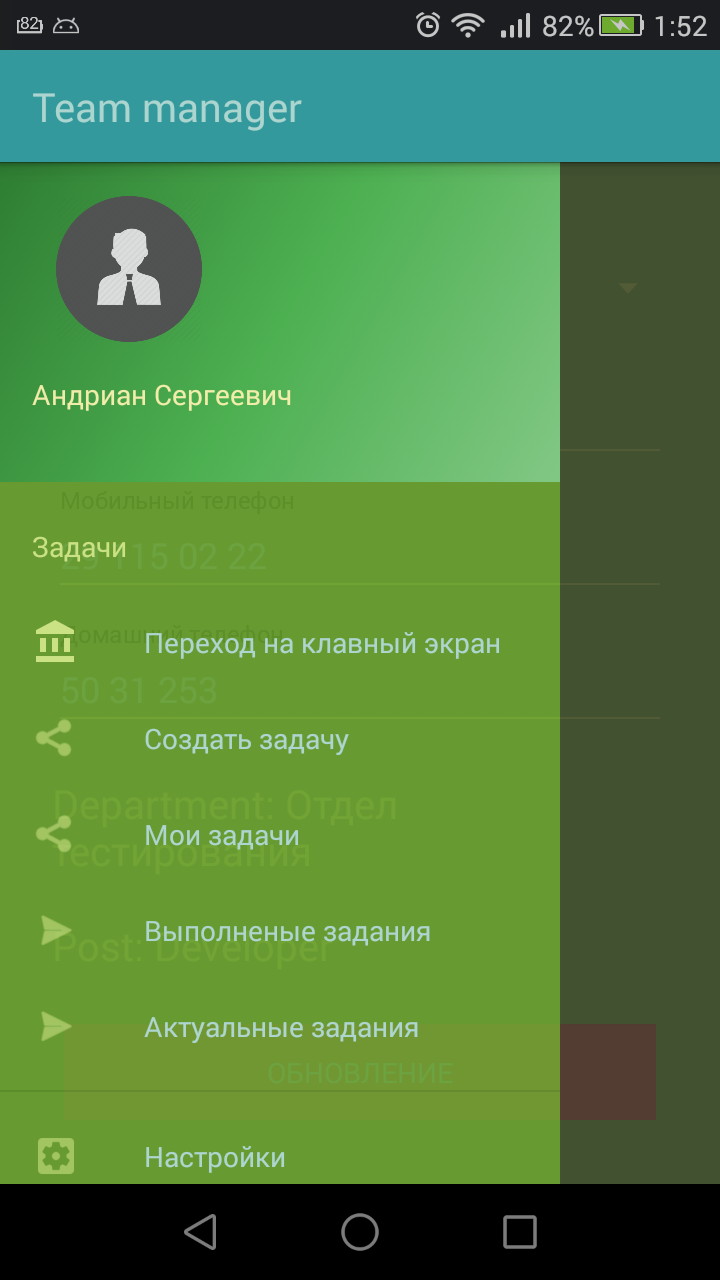


Рисунок 3.13 – Экран личного кабинета

На рисунке 3.14, изображён экран добавления задачи в систему. Все поля проверяются на наполненность информации и их валидация. Таким образом система защищена от неверных данных.

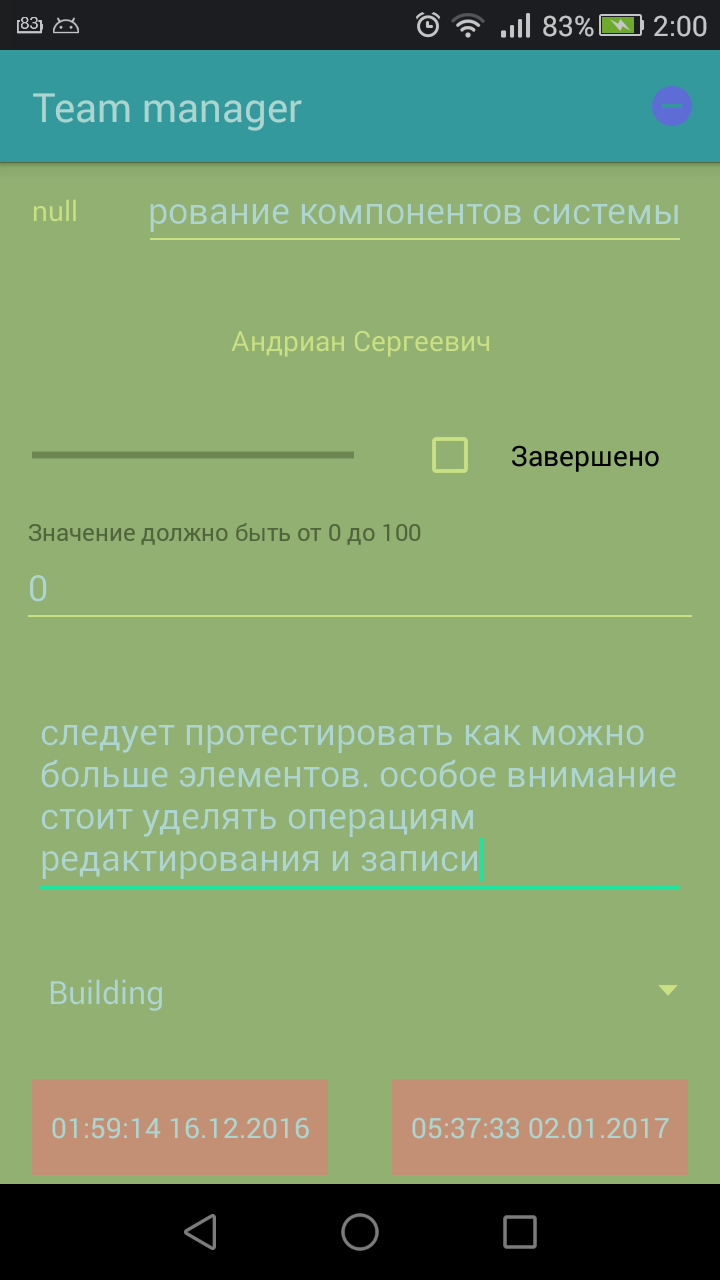


Рисунок 3.14 – Экран добавления задачи

На рисунке 3.15 показано, что при успешном добавлении задачи, выводится соответствующее сообщение, которое позволяет уведомить пользователя о результате операции добавления задачи. Если при создании возникает ошибка, то пользователь об этом уведомляется, а если ошибка вызвана системой, то будет предложено отправить отчет об ошибке, который генерирует библиотека автоматического сбора ошибок ACRA.

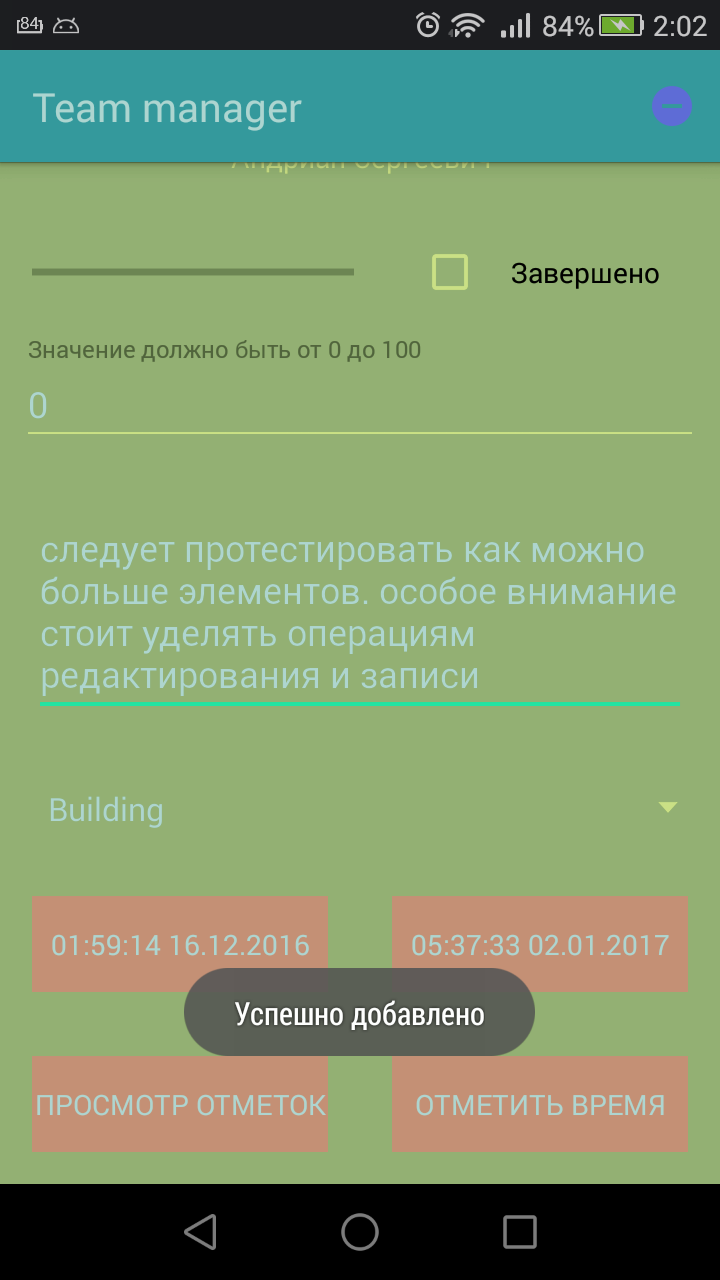


Рисунок 3.15 – Сообщение об успешном добавлении задачи

Нажмем на кнопку «отметить время», в результате приложение перейдет к новому экрану, изображённому на рисунке 3.16, на котором будет новая форма, в которой можно заполнить информацию о активности над задачей и добавить её в базу. При переходе на экран в котором происходит отметка времени, на экране отображается название задачи, тем самым, пользователь понимает в какую задачу он отмечает время. При успешном добавлении, так же выводится сообщение об успешном добавлении, если какая-то ошибка, пользователь будет уведомлен об этом.

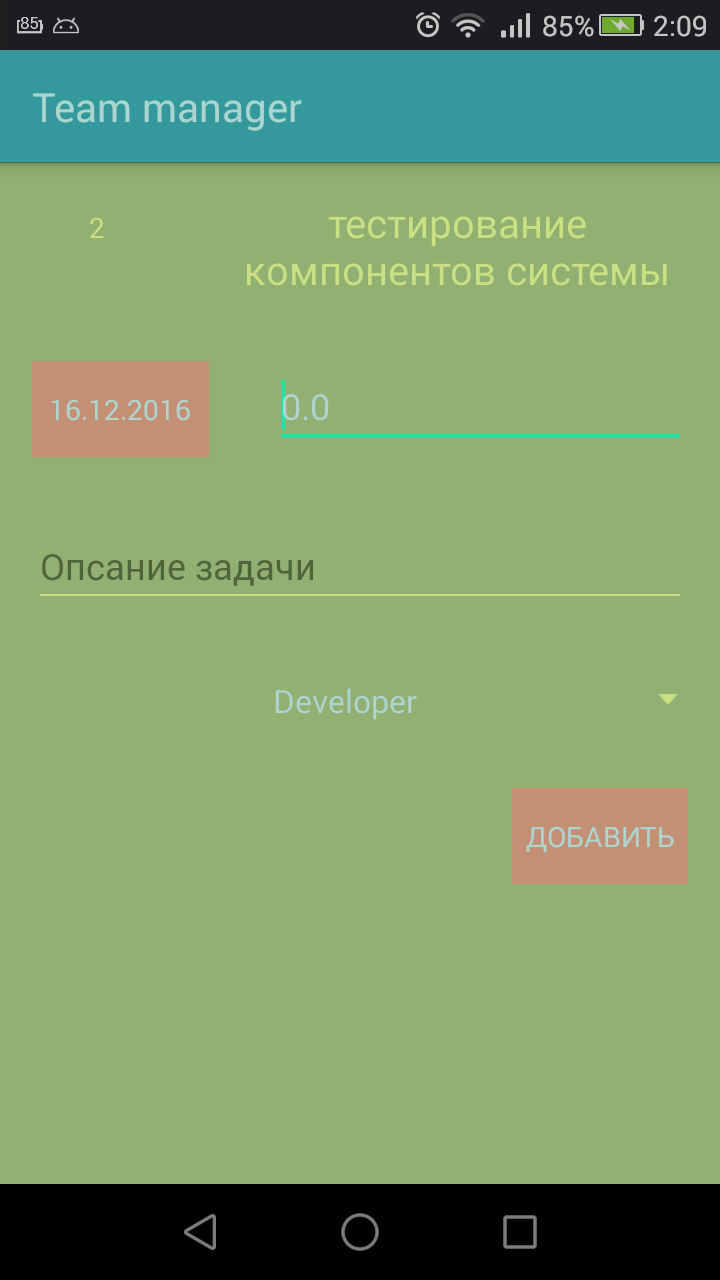


Рисунок 3.16 – Экран добавления отметки времени

Теперь просмотрим лог по задаче, которую мы только что создали. В результате, отображённом на рисунке 3.17 списке логов, мы увидим единственную нашу отметку.

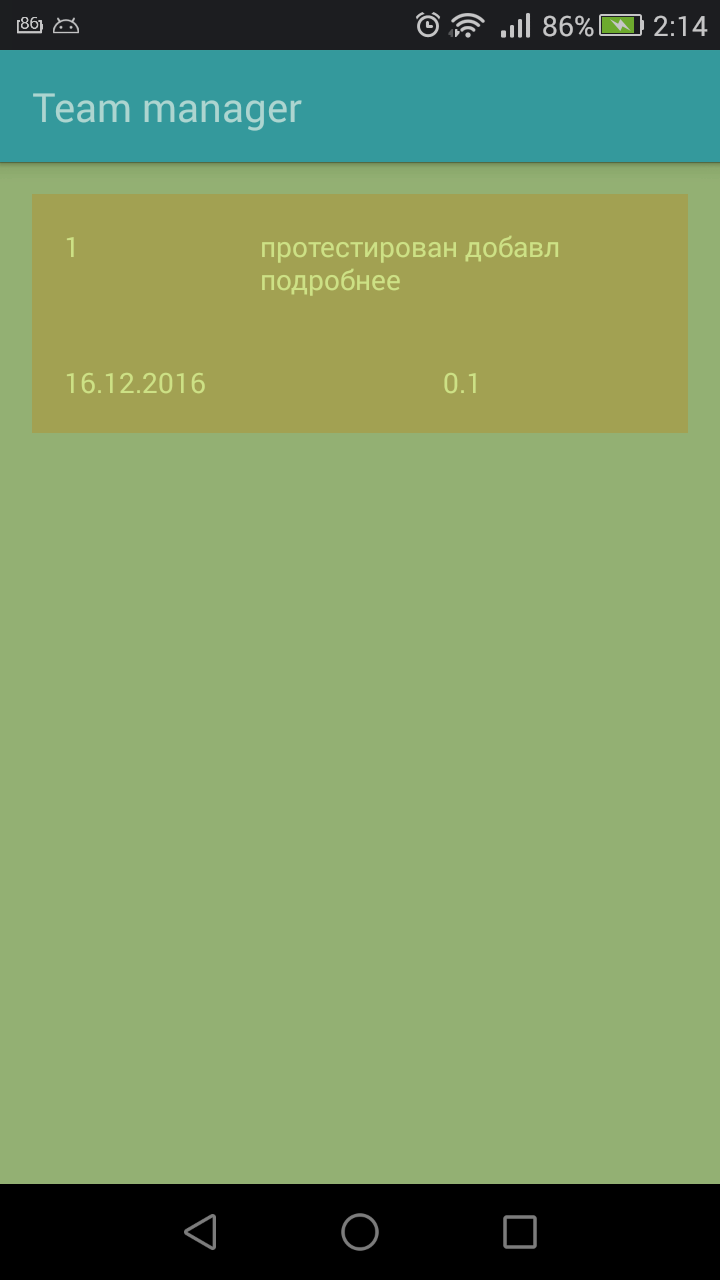


Рисунок 3.17 – Экран отображения спискового представления логов по определенной задаче

Теперь присоединим ещё одного работника к нашей задаче. Для этого необходимо вернутся на экран нашей задачи как на рисунке 3.18 и выбрать действие «присоединить». В результате откроется новая активность со списком пользователей в системе, которых можно присоединить к задаче.

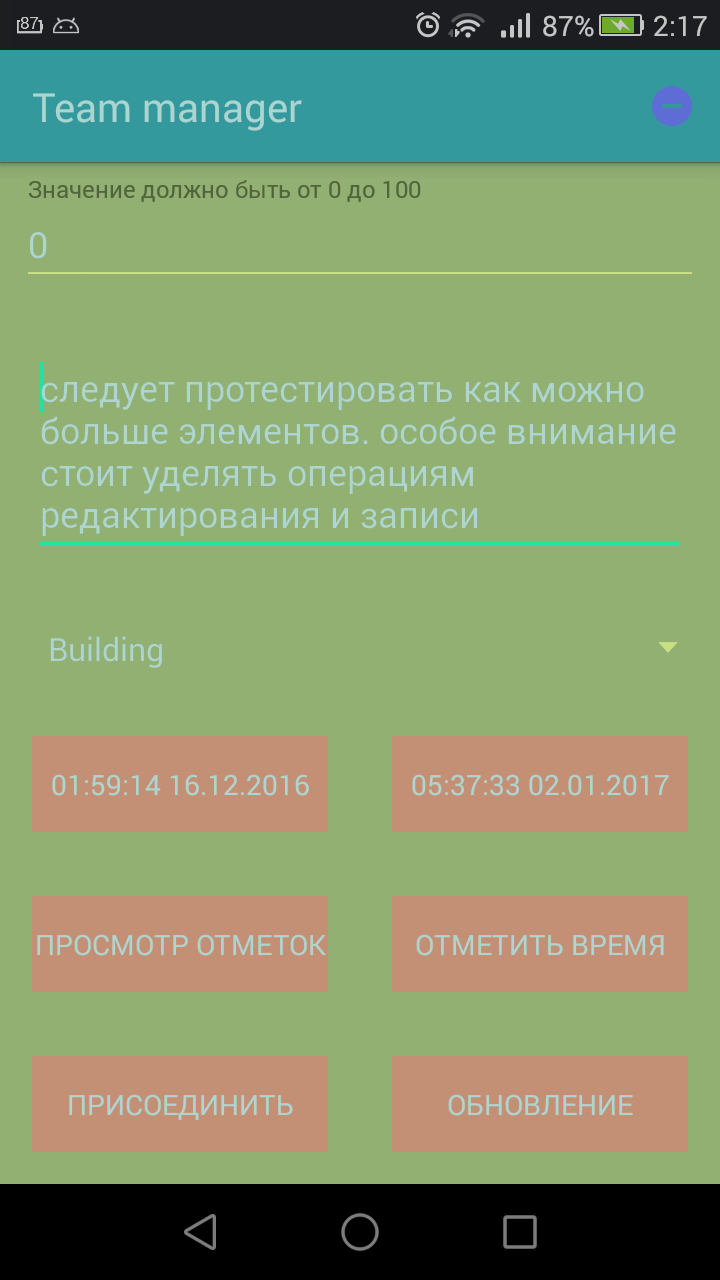


Рисунок 3.18 – Экран заполненной задачи

На рисунке 3.19 отображен список пользователей, которые зарегистрированы в системе. Для каждого элемента списка был создан отдельно макет, который отображает фамилию, имя, отчество, отдел и должность человека. Для присоединения человека к задаче необходимо нажать на него и подержать около секунды фокус на этом элементе. В результате появится снизу элемент, который просит подтверждения намерению добавить пользователя к проекту.

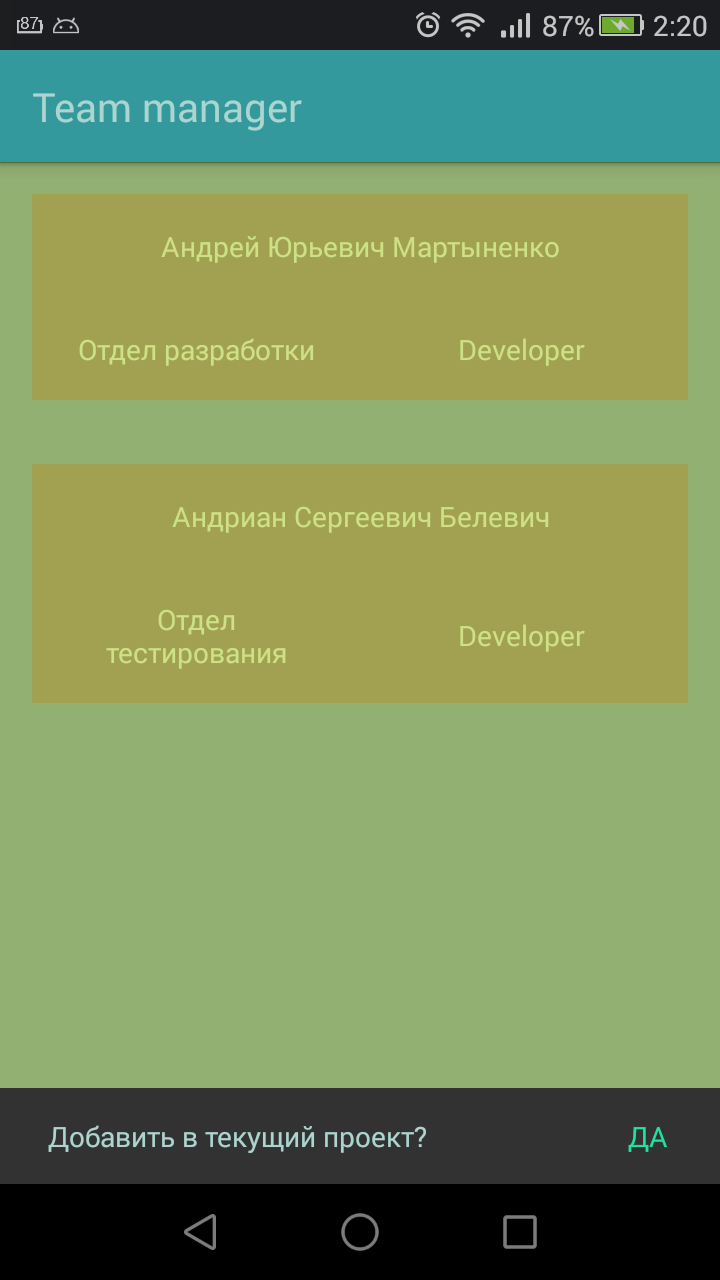


Рисунок 3.19 – Список пользователей и элемент подтверждения действия

Как в случае и с задачами и логами, тут тоже идёт уведомление пользователя о успешном, либо неуспешном действии добавления.

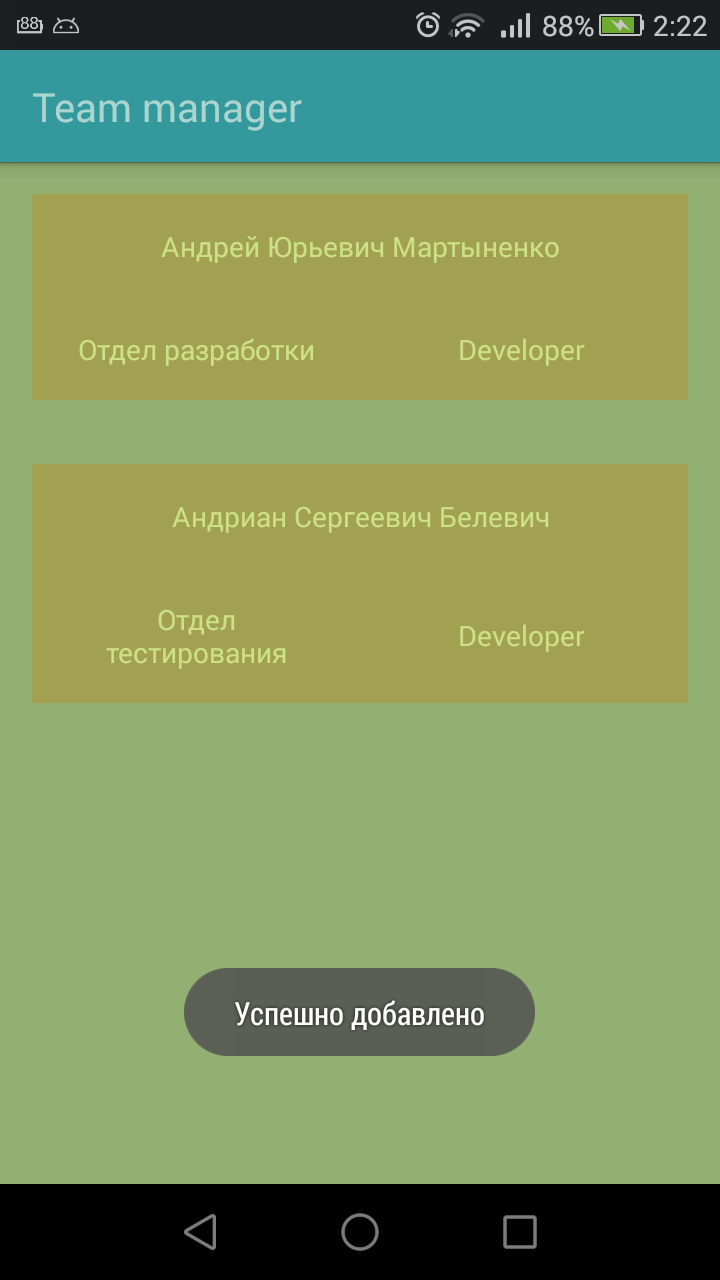


Рисунок 3.20 – Успешное добавление пользователя к проекту

ЗАКЛЮЧЕНИЕ

В результате решения поставленных задач, получилось полноценное мобильное приложение, с реализованными функциональными возможностями, позволяющее управлять временем в своих задачах, а также работать над задачами совместно с другими пользователями. Трекинговая система позволяет отслеживать время, которое уходит на конкретно задачу и позволяет пользователю рассматривать и анализировать потраченное время.

Для успешной реализации поставленных задач был произведен анализ методов управления временем и определены основные функции разрабатываемой системы.

На основе предметной области были спроектированы сущности представления и описана информационная модель приложения.

Так же следует отметить, что были описаны и реализованы алгоритмы основных функций приложения.

В результате, была достигнута цель, которая была поставлена перед данным курсовым проектом – автоматизация контроля временем. Теперь не надо искать в ежедневнике или на листиках по задачам время. Приложение позволяет просмотреть все быстро и в понятной форме.

За счет доступного, простого и приятного интерфейса, обеспечивается снижение сложностей с использованием данной системы и допуском ошибок.

Данное приложение можно улучшать, вынеся всю логику сохранения информации в облачные сервисы, или разработки собственного REST full сервера, на котором будет храниться информация, что позволить множеству пользователей работать удаленно и отмечать успехи по совместным задачам. Так же следует отметить, что можно добавить службы push-уведомлений приложения о настигающих просроченных задачах. Так же для удобства работы и точного контроля времени, на экран отчета активности по задаче, можно поместить элемент, который будет запускать фоновую задачу, которая будет отсчитывать время, которое потратил пользователь на задачу. Это уточнит подсчет времени и у работников будет меньше мотивации отвлекаться, так как будет меньше способов избежать наказания, за бездеятельность.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Mental Skills [Электронный ресурс]. – Электронные данные. – Режим доступа : <http://www.mental-skills.ru/dict/upravlenie-vremenem-taym-menedzhment/>

[2] 4Brain [Электронный ресурс]. – Электронные данные. – Режим доступа : <https://4brain.ru/time/>

[3] Android. Программирование для профессионалов. 2-е издание / Харди Б., Филлипс Б., Стюарт К., Марсикано К. – Питер 2016 – 730 с.

[4] Горбачев, А. А. Тайм-менеджмент в два счета / А. А. Горбачев – «Питер» - С-Пб, 2009. – 256 с.

[5] Верещагин, А. А. Как все успевать? 4 навыка эффективного управления временем. / А. А. Верещагин – «Весь» - С-Пб, 2016 – 214 с.

[6] Тайм-менеджмент [Электронный ресурс]. – Электронные данные. – Режим доступа : [https://4brain.ru/blog/](https://4brain.ru/blog/%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0-%D1%8D%D0%B9%D0%B7%D0%B5%D0%BD%D1%85%D0%B0%D1%83%D1%8D%D1%80%D0%B0/)

[7] Конструктор успеха [Электронный ресурс]. – Электронные данные. – Режим доступа : <http://constructorus.ru/uspex/zakon-pareto.html>

[8] Современные средства управлением временем [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://timefly.helllab.ru/time-management/techniques>

[9] Worksection [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://worksection.com/blog/time-management.html>

[10] Developer.android.com [Электронный ресурс] – Электронные данные – Режим доступа: <https://developer.android.com/index.html>

[11] It-gost [Электронный ресурс] – Электронные данные – Режим доступа: http://it-gost.ru/articles/view\_articles/94

Приложение А

(обязательное)

Функциональная модель «TO BE» (к пункту 3.3)



Рисунок А.1 – Контекстная диаграмма процесса автоматизации



Рисунок А.2 – Декомпозиция процесса автоматизации планирования рабочего времени

Продолжение приложения А



Рисунок А.3 – Декомпозиция процесса подготовки к работе



Рисунок А.4 – Декомпозиция процесса анализа результатов

Приложение Б

(обязательное)

Cхема функции (к пункту 3.6)



Рисунок Б.1 – схема алгоритма добавления отчета времени в задачу

Приложение В

(обязательное)

Листинг кода основных элементов программы

package by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.sqllite;  
  
import android.content.ContentValues;  
import android.database.Cursor;  
import android.database.CursorWrapper;  
import android.database.sqlite.SQLiteDatabase;  
import android.util.Log;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.BaseHelper;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.cursorwrapper.BaseCustomCursorWrapper;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.entity.Entity;  
  
/\*\*  
 \* Created by Andrey on 29.11.2016.  
 \*/  
  
abstract class AbstractDaoBase<T extends Entity>{  
  
 private static final String TAG = AbstractDaoBase.class.getName();  
 private static final String MSG\_ERROR\_READ = "Error read to database ";  
 private static final String SQL\_SELECT\_ALL\_QUERY = "SELECT \* FROM ";  
  
 protected abstract ContentValues getContentValuesNotId(T value);  
 protected abstract ContentValues getContentValuesWithId(T value);  
 protected abstract BaseCustomCursorWrapper<T> getCursorWrapper(Cursor cursor);  
  
 private SQLiteDatabase mDatabase;  
  
 protected AbstractDaoBase(BaseHelper mDatabase) {  
  
 this.mDatabase = mDatabase.getWritableDatabase();  
 }  
  
 protected AbstractDaoBase(SQLiteDatabase mDatabase){  
  
 this.mDatabase = mDatabase;  
 }  
  
  
 final protected long create(T entity, final String NAME\_TABLE) {  
  
 ContentValues values = getContentValuesNotId(entity);  
  
 return mDatabase.insertOrThrow(NAME\_TABLE, null, values);  
 }  
  
 final protected List<T> create(List<T> entities, final String NAME\_TABLE){  
  
 for(T entity: entities){

Продолжение приложения В

create(entity, NAME\_TABLE);  
 }  
  
 return entities;  
 }  
  
  
 final protected T read(final String NAME\_TABLE,  
 final String WHERE\_CLAUSE,  
 final String[] WHERE\_ARG) {  
  
 Entity result = null;  
  
 try (BaseCustomCursorWrapper cursorWrapper =  
 getCursorWrapper(queryCrimesWhere(NAME\_TABLE, null, WHERE\_CLAUSE, WHERE\_ARG, null))) {  
  
 if (cursorWrapper.getCount() != 0) {  
  
 cursorWrapper.moveToFirst();  
 result = cursorWrapper.getData();  
 }  
 } catch (Exception e){  
 Log.e(TAG, MSG\_ERROR\_READ, e);  
 }  
  
 return (T) result;  
 }  
  
 final public List<T> reads(final String NAME\_TABLE) {  
  
 List<Entity> tList = new ArrayList<>();  
  
 try(BaseCustomCursorWrapper cursorWrapper =  
 getCursorWrapper(mDatabase.rawQuery(SQL\_SELECT\_ALL\_QUERY+ NAME\_TABLE, null))){  
  
 if(cursorWrapper.moveToFirst()){  
 do {  
 // Adding to list  
 tList.add(cursorWrapper.getData());  
 } while (cursorWrapper.moveToNext());  
 }  
  
 }  
  
 return (List<T>) tList;  
 }  
  
 final public List<T> reads(final String NAME\_TABLE, final String WHERE\_CAUSE, final String[] ARG\_WHERE) {  
  
 List<Entity> tList = new ArrayList<>();  
  
 try(BaseCustomCursorWrapper cursorWrapper =  
 getCursorWrapper(queryCrimesWhere(NAME\_TABLE, null, WHERE\_CAUSE, ARG\_WHERE, null))){

Продолжение приложения В

if(cursorWrapper.moveToFirst()){  
 do {  
 // Adding to list  
 tList.add(cursorWrapper.getData());  
 } while (cursorWrapper.moveToNext());  
 }  
  
 }  
  
 return (List<T>) tList;  
 }  
  
  
 final protected T update(T entity,  
 final String NAME\_TABLE,  
 final String WHERE\_CLAUSE,  
 final String[] WHERE\_ARG) {  
  
 ContentValues values = getContentValuesNotId(entity);  
  
 mDatabase.update(NAME\_TABLE, values,  
 WHERE\_CLAUSE,  
 WHERE\_ARG);  
  
 Log.d(TAG, "update entity = " + entity.toString());  
  
 return entity;  
  
 }  
  
  
 final protected int delete(final String NAME\_TABLE, final String WHERE\_CLAUSE, final String[] WHERE\_ARG) {  
  
 return mDatabase.delete(NAME\_TABLE, WHERE\_CLAUSE, WHERE\_ARG);  
 }  
  
 final protected int deleteAll(final String NAME\_TABLE){  
  
 return delete(NAME\_TABLE, null, null);  
 }  
 private CursorWrapper queryCrimesWhere(final String NAME\_TABLE,  
 final String[] COLUMNS,  
 final String WHERE\_CLAUSE,  
 final String[] WHERE\_ARG,  
 final String ORDER) {  
  
 Cursor cursor = mDatabase.query(  
 NAME\_TABLE, //name table  
 COLUMNS, // Columns - null выбирает все столбцы  
 WHERE\_CLAUSE, //where if  
 WHERE\_ARG,//if args  
 null, // groupBy  
 null, // having  
 ORDER // orderBy  
 );  
  
 return new CursorWrapper(cursor);

Продолжение приложения В

}  
  
}

package by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db;  
  
import android.content.Context;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
import android.util.Log;  
  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.CityDao;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.CountryDao;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.DepartmentDao;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.PersonDao;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.PostDao;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.TypeActivityDao;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.TypeTaskDao;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.TypeUserDao;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.UserDao;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.sqllite.CityDaoLite;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.sqllite.CountryDaoLite;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.sqllite.DepartmentDaoLite;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.sqllite.PersonDaoLite;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.sqllite.PostDaoLite;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.sqllite.TypeActivityDaoLite;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.sqllite.TypeTaskDaoLite;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.sqllite.TypeUserDaoLite;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.dao.sqllite.UserDaoLite;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.startdata.CityStorageInit;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.startdata.CountryStorageInit;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.startdata.DepartmentStorageInit;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.startdata.PersonStorageInit;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.startdata.PostStorageInit;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.startdata.TypeActivityStorageInit;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.startdata.TypeTaskSorageInit;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.startdata.TypeUserStorageInit;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.startdata.UserStorageInit;  
  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.CityTable;  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.CountryTable

import static

Продолжение приложения В

by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.DepartmentTable;  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.HasTaskPersonTable;  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.LogTimeTaskTable;  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.PersonTable;  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.PostTable;  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.TaskTable;  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.TypeActivityTable;  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.TypeTaskTable;  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.TypeUserTable;  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.UserTable;  
  
/\*\*  
 \* Created by Andrey on 27.11.2016.  
 \*/  
  
abstract public class BaseHelper extends SQLiteOpenHelper {  
  
 private static final String TAG = BaseHelper.class.getName();  
  
 protected static Context context;  
  
 protected BaseHelper(Context context,  
 final String DATABASE\_NAME,  
 final SQLiteDatabase.CursorFactory cursorFactory,  
 final int VERSION) {  
  
 super(context, DATABASE\_NAME, cursorFactory, VERSION);  
 }  
  
 public void close(){  
 super.close();  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
  
 db.execSQL("CREATE TABLE " + CountryTable.NAME + " (" +  
 CountryTable.Colums.ID\_COUNTRY +  
 " INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 CountryTable.Colums.NAME\_COUNTRY + " TEXT, " +  
 CountryTable.Colums.KEY\_CONTRY + " TEXT, " +  
 CountryTable.Colums.KEY\_PHONE +" TEXT " +  
 ")");  
  
 Log.d(TAG, "create table (country successfully)");

Продолжение приложения В

db.execSQL("CREATE TABLE " + CityTable.NAME + " (" +  
 CityTable.Colums.ID\_CITY +  
 " INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 CityTable.Colums.NAME + " TEXT, " +  
 CityTable.Colums.CODE\_PHONE + " TEXT, " +  
 CityTable.Colums.ID\_COUNTRY + " INTEGER NOT NULL, " +  
 CityTable.REF\_COUNTRY +  
 ")");  
  
 Log.d(TAG, "create table (city successfully)");  
  
 db.execSQL("CREATE TABLE " + DepartmentTable.NAME + " (" +  
 DepartmentTable.Colums.ID\_DEPARTMENT +  
 " INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 DepartmentTable.Colums.NAME\_DEPARTMENT + " TEXT "+  
 ")");  
  
 Log.d(TAG, "create table (department successfully)");  
  
 db.execSQL("CREATE TABLE " + PostTable.NAME + " (" +  
 PostTable.Colums.ID\_POST +  
 " INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 PostTable.Colums.NAME\_POST + " TEXT NOT NULL, " +  
 PostTable.Colums.RATE + " FLOAT"+  
 ")");  
  
 Log.d(TAG, "create table (post successfully)");  
  
 db.execSQL("CREATE TABLE " + TypeUserTable.NAME + " (" +  
 TypeUserTable.Colums.ID\_TYPE +  
 " INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 TypeUserTable.Colums.NAME + " TEXT NOT NULL, " +  
 TypeUserTable.Colums.ACCESS\_LEVEL + " INTEGER NOT NULL"+  
 ")");  
  
 Log.d(TAG, "create table (type\_user successfully)");  
  
 db.execSQL("CREATE TABLE " + UserTable.NAME + " (" +  
 UserTable.Colums.ID\_USER +  
 " INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 UserTable.Colums.LOGIN + " TEXT NOT NULL, " +  
 UserTable.Colums.PASSWORD + " TEXT NOT NULL, " +  
 UserTable.Colums.ID\_TYPE\_USER + " INTEGER NOT NULL, " +  
 UserTable.REF\_TYPE\_USER +  
 ")");  
  
 Log.d(TAG, "create table (user successfully)");  
  
 db.execSQL("CREATE TABLE " + PersonTable.NAME + " (" +  
 PersonTable.Colums.ID\_PERSON +  
 " INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 PersonTable.Colums.SURNAME + " TEXT, " +  
 PersonTable.Colums.NAME + " TEXT, " +  
 PersonTable.Colums.PATRONYMIC + " TEXT, " +  
 PersonTable.Colums.DATE\_OF\_BIRTH + " TEXT, " +  
 PersonTable.Colums.SEX + " TEXT, " +  
 PersonTable.Colums.E\_MAIL + " TEXT, " +  
 PersonTable.Colums.ID\_USER + " INTEGER NOT NULL, " +

Продолжение приложения В

PersonTable.Colums.ID\_POST + " INTEGER NOT NULL, " +  
 PersonTable.Colums.ID\_DEPARTMENT + " INTEGER NOT NULL, " +  
 PersonTable.Colums.ID\_CITY + " INTEGER NOT NULL, " +  
 PersonTable.Colums.MOBILE\_PHONE + " TEXT, " +  
 PersonTable.Colums.HOME\_PHONE + " TEXT, " +  
 PersonTable.REF\_USER + ',' +  
 PersonTable.REF\_POST + ',' +  
 PersonTable.REF\_DEPARTMENT + ',' +  
 PersonTable.REF\_CITY +  
 ")");  
  
 Log.d(TAG, "create table (person successfully)");  
  
 db.execSQL("CREATE TABLE " + TypeTaskTable.NAME + " (" +  
 TypeTaskTable.Colums.ID\_TYPE\_TASK +  
 " INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 TypeTaskTable.Colums.NAME\_TYPE + " TEXT NOT NULL"+  
 ")");  
  
 Log.d(TAG, "create table (type\_task successfully)");  
  
 db.execSQL("CREATE TABLE " + TaskTable.NAME + " (" +  
 TaskTable.Colums.ID\_TASK + " INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 TaskTable.Colums.NAME + " TEXT, " +  
 TaskTable.Colums.DESCRIPTION + " TEXT, " +  
 TaskTable.Colums.DATE\_BEGIN + " TEXT, " +  
 TaskTable.Colums.DATE\_END + " TEXT, " +  
 TaskTable.Colums.DONE + " INTEGER, " +  
 TaskTable.Colums.PROGRESS + " INTEGER, " +  
 TaskTable.Colums.ID\_TYPE\_TASK + " INTEGER NOT NULL, " +  
 TaskTable.Colums.ID\_PERSON\_ADD + " INTEGER NOT NULL, " +  
 TaskTable.REF\_TYPE\_TASK + ',' +  
 TaskTable.REF\_PERSON\_ADD +  
 ")");  
  
 Log.d(TAG, "create table (task successfully)");  
  
 db.execSQL("CREATE TABLE " + HasTaskPersonTable.NAME + " (" +  
 HasTaskPersonTable.Colums.ID\_HAS\_TASK\_PERSON +  
 " INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 HasTaskPersonTable.Colums.ID\_TASK + " INTEGER NOT NULL, " +  
 HasTaskPersonTable.Colums.ID\_PERSON + " INTEGER NOT NULL, " +  
 HasTaskPersonTable.REF\_PERSON + ',' +  
 HasTaskPersonTable.REF\_TASK +  
 ")");  
  
 Log.d(TAG, "create table (has\_task\_person successfully)");  
  
 db.execSQL("CREATE TABLE " + TypeActivityTable.NAME + " (" +  
 TypeActivityTable.Colums.ID\_TYPE\_ACTIVITY + " INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 TypeActivityTable.Colums.NAME\_ACTIVITY + " TEXT NOT NULL"+  
 ")");  
  
 Log.d(TAG, "create table (type\_activity successfully)");  
  
 db.execSQL("CREATE TABLE " + LogTimeTaskTable.NAME + " (" +

Продолжение приложения В

LogTimeTaskTable.Colums.ID\_LOG + " INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, " +  
 LogTimeTaskTable.Colums.DATE\_LOG + " TEXT NOT NULL, " +  
 LogTimeTaskTable.Colums.DESCRIPTION + " TEXT, " +  
 LogTimeTaskTable.Colums.HOURS + " FLOAT, " +  
 LogTimeTaskTable.Colums.LINKS\_EXT\_STOR + " TEXT, " +  
 LogTimeTaskTable.Colums.ID\_TYPE\_ACTIVITY + " INTEGER NOT NULL, " +  
 LogTimeTaskTable.Colums.ID\_HAS\_TASK\_PERSON + " INTEGER NOT NULL, " +  
 LogTimeTaskTable.REF\_TYPE\_ACTIVITY + ',' +  
 LogTimeTaskTable.REF\_HAS\_TASK\_PERSON +  
 ")");  
  
 Log.d(TAG, "create table (log\_time\_task successfully)");  
  
 Log.d(TAG, "create Base (successfully");  
  
 putDataBase(db);  
  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
  
 }  
  
 @Override  
 public void onConfigure(SQLiteDatabase db) {  
 super.onConfigure(db);  
 db.setForeignKeyConstraintsEnabled(true);  
 }  
  
 private void putDataBase(SQLiteDatabase db){  
 CountryDao countryDao = new CountryDaoLite(db);  
 countryDao.create(new CountryStorageInit().getData());  
  
 CityDao cityDaoLite = new CityDaoLite(db);  
 cityDaoLite.create(new CityStorageInit().getData(countryDao.reads()));  
  
 DepartmentDao departmentDaoLite = new DepartmentDaoLite(db);  
 departmentDaoLite.create(new DepartmentStorageInit().getData());  
  
 PostDao postDaoLite = new PostDaoLite(db);  
 postDaoLite.create(new PostStorageInit().getData());  
  
 TypeActivityDao typeActivityDaoLite = new TypeActivityDaoLite(db);  
 typeActivityDaoLite.create(new TypeActivityStorageInit().getData());  
  
 TypeTaskDao typeTaskDaoLite = new TypeTaskDaoLite(db);  
 typeTaskDaoLite.create(new TypeTaskSorageInit().getData());  
  
 TypeUserDao typeUserDaoLite = new TypeUserDaoLite(db);  
 typeUserDaoLite.create(new TypeUserStorageInit().getData());  
  
 UserDao userDao = new UserDaoLite(db);  
 userDao.create(new UserStorageInit().getData(typeUserDaoLite.reads()));  
  
 PersonDao personDao = new PersonDaoLite(db);  
 personDao.create(new PersonStorageInit().getData

Продолжение приложения В

userDao.reads(),  
 postDaoLite.reads(),  
 departmentDaoLite.reads(),  
 cityDaoLite.reads()  
 ));  
  
 }  
}

package by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.cursorwrapper;  
  
import android.database.Cursor;  
  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.entity.CityEntity;  
  
import static by.bsuir.zavadatar.andrey.teammanagerbsuir.model.db.KorpPortalDBSchema.\*;  
  
/\*\*  
 \* Created by Andrey on 27.11.2016.  
 \*/  
  
public class CityCursorWrapper extends BaseCustomCursorWrapper<CityEntity> {  
  
 /\*\*  
 \* Creates a cursor wrapper.  
 \*  
 \* @param cursor The underlying cursor to wrap.  
 \*/  
 public CityCursorWrapper(Cursor cursor) {  
 super(cursor);  
 }  
  
 @Override  
 public CityEntity getData(){  
  
 CityEntity cityEntity = new CityEntity();  
  
 cityEntity.setIdCity(getInt(getColumnIndex(CityTable.Colums.ID\_CITY)));  
 cityEntity.setName(getString(getColumnIndex(CityTable.Colums.NAME)));  
 cityEntity.setCodePhone(getString(getColumnIndex(CityTable.Colums.CODE\_PHONE)));  
 cityEntity.setIdCountry(getInt(getColumnIndex(CityTable.Colums.ID\_COUNTRY)));  
  
 return cityEntity;  
 }  
}

package by.bsuir.zavadatar.andrey.teammanagerbsuir.controllers.activity;  
  
  
import android.content.Intent;  
import android.graphics.BitmapFactory;  
import android.os.Bundle;  
import android.support.annotation.NonNull;  
import android.support.annotation.Nullable;  
import android.support.design.widget.NavigationView;  
import android.support.v4.app.Fragment;

Продолжение приложения В

import android.support.v4.app.FragmentManager;  
import android.support.v4.view.GravityCompat;  
import android.support.v4.widget.DrawerLayout;  
import android.support.v7.app.AppCompatActivity;  
import android.util.Log;  
import android.view.MenuItem;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.controllers.TypeShowTaskList;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.controllers.enumiration.PersonShowFilter;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.model.entity.enumiration.TypeUserName;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.storage.ApplicationSettings;  
import by.bsuir.zavadatar.andrey.teammanagerbsuir.utils.Converter;  
  
public abstract class PersonUseApplicationActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener{  
  
 private static final String TAG = PersonUseApplicationActivity.class.getName();  
 private NavigationView navigationView;  
 private DrawerLayout mDrawerLayout;  
 private FragmentManager fm;  
  
 private TextView mTvNamePerson;  
 private ImageView mIvPhotoPerson;  
  
 protected abstract Fragment createFragment();  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.person\_room\_application);  
  
 if(!ApplicationSettings.isAuthorisation(this)){  
 gotoLoginPage();  
 }  
  
 mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer\_person\_room\_application);  
  
 navigationView = (NavigationView) findViewById(R.id.nav\_view\_person\_room\_application);  
 navigationView.setNavigationItemSelectedListener(this);  
  
 int menuResID;  
 if(ApplicationSettings.getAccessLevelName(getApplicationContext()).equals(TypeUserName.Admin))  
 menuResID = R.menu.admin\_room\_drawer;  
 else  
 menuResID = R.menu.user\_room\_drawer;  
 navigationView.inflateMenu(menuResID);  
  
 mTvNamePerson = (TextView) navigationView.getHeaderView(0).findViewById(R.id.name\_person\_nav\_panel);  
 mTvNamePerson.setText(ApplicationSettings.getFIO(this))

Продолжение приложения В

mIvPhotoPerson = (ImageView) navigationView.getHeaderView(0).findViewById(R.id.image\_person);  
 BitmapFactory.Options options = new BitmapFactory.Options();  
 options.outHeight = (int) Converter.convertDpToPixel(40, getApplicationContext());  
 options.outWidth = (int) Converter.convertDpToPixel(40, getApplicationContext());  
 mIvPhotoPerson.setImageBitmap(BitmapFactory.decodeResource(getResources(), R.drawable.user, options));  
  
 fm = getSupportFragmentManager();  
 Fragment fragment = fm.findFragmentById(R.id.frame\_person\_room\_application);  
  
 if (fragment == null) {  
 fragment = createFragment();  
 if(fragment != null)  
 fm.beginTransaction()  
 .replace(R.id.frame\_person\_room\_application, fragment)  
 .commit();  
 }  
  
 Log.d(TAG, "create PersonUseApplicationActivity");  
 }  
  
 final protected void replaceFrame(@NonNull Fragment frame){  
 fm.beginTransaction()  
 .replace(R.id.frame\_person\_room\_application, frame)  
 .commit();  
 }  
  
 private void gotoLoginPage() {  
 finish();  
 startActivity(LoginActivity.newIntent(this));  
 }  
  
 @SuppressWarnings("StatementWithEmptyBody")  
 @Override  
 public boolean onNavigationItemSelected(@Nullable MenuItem item) {  
 // Handle navigation view item clicks here.  
 int id = item.getItemId();  
  
 switch (id){  
 //User menu item  
 case R.id.go\_to\_home\_person:{  
 finish();  
 startActivity(UserRoomActivity.newIntent(getApplicationContext()));  
 } break;  
 case R.id.nav\_add\_task\_this\_person:{  
 startActivity(TaskSingleActivity.newIntent(getApplicationContext()));  
 } break;  
 case R.id.nav\_all\_tasks\_this\_person:{  
  
 startActivity(TaskListFragmentActivity.newInstance(getApplicationContext(), TypeShowTaskList.SHOW\_All\_PERSON\_TASK));  
 } break;  
 case R.id.nav\_done\_tasks\_this\_person:{\ startActivity(TaskListFragmentActivity.newInstance(getApplicationContext(),

Продолжение приложения В

TypeShowTaskList.SHOW\_DONE\_PERSON\_TASK));  
 } break;  
 case R.id.nav\_current\_tasks\_this\_person:{  
  
 startActivity(TaskListFragmentActivity.newInstance(getApplicationContext(), TypeShowTaskList.SHOW\_CURRENT\_PERSON\_TASK));  
 } break;  
 case R.id.setting\_application\_person:{  
 finish();  
 startActivity(new Intent(getApplicationContext(), SettingApplicationActivity.class));  
 } break;  
 case R.id.lot\_out\_person:{  
 ApplicationSettings.logOut(getApplicationContext());  
 finish();  
 startActivity(new Intent(LoginActivity.newIntent(getApplicationContext())));  
 } break;  
 //Admin menu item  
 case R.id.nav\_show\_all\_tasks\_person:{  
 startActivity(TaskListFragmentActivity.newInstance(getApplicationContext(), TypeShowTaskList.SHOW\_ALL\_TASK));  
 } break;  
 case R.id.show\_person:{  
 startActivity(PersonListActivity.newInstance(this, PersonShowFilter.ADMIN\_SHOW));  
 } break;  
 case R.id.show\_users:{  
 startActivity(UserListFragmentActivity.newInstance(this));  
 } break;  
  
 default:  
 Log.d(TAG, "Default Action (no action)");  
 }  
  
 mDrawerLayout.closeDrawer(GravityCompat.START);  
 return true;  
 }  
}

Приложение Г

(обязательное)

Листинг скрипта создания базы данных

CREATE TABLE country (

id\_country INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

name\_country TEXT ,

key\_contry TEXT ,

key\_phone TEXT

)

CREATE TABLE city (

id\_city INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

name TEXT,

code\_phone TEXT,

id\_country INTEGER NOT NULL,

FOREIGN KEY (id\_country) REFERENCES country(id\_country)

)

CREATE TABLE department (

id\_department INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

name\_department TEXT

)

CREATE TABLE post (

id\_post INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

name\_post TEXT NOT NULL,

rate FLOAT

)

CREATE TABLE type\_user (

id\_type INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

name TEXT NOT NULL,

access\_level INTEGER NOT NULL

)

CREATE TABLE user (

id\_user INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

login TEXT NOT NULL,

password TEXT NOT NULL,

id\_type\_user INTEGER NOT NULL,

FOREIGN KEY (id\_type\_user) REFERENCES type\_user(id\_type)

)

CREATE TABLE person (

id\_person INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

surname TEXT ,

Продолжение приложения Г

name TEXT ,

patronymic TEXT,

date\_of\_birth DATE,

sex TEXT,

e\_mail TEXT,

id\_user INTEGER NOT NULL,

id\_post INTEGER NOT NULL,

id\_department INTEGER NOT NULL,

id\_city INTEGER NOT NULL,

mobile\_phone TEXT,

home\_phone TEXT,

FOREIGN KEY (id\_user) REFERENCES user(id\_user),

FOREIGN KEY (id\_post) REFERENCES post(id\_post),

FOREIGN KEY (id\_department) REFERENCES department(id\_department),

FOREIGN KEY (id\_city) REFERENCES city(id\_city)

)

CREATE TABLE type\_task (

id\_type\_task INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

name\_type TEXT NOT NULL

)

CREATE TABLE task (

id\_task INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,

name TEXT ,

description TEXT,

date\_begin DATE ,

date\_end DATE,

done INTEGER,

progress INTEGER,

id\_type\_task INTEGER NOT NULL,

id\_person\_add INTEGER NOT NULL,

FOREIGN KEY (id\_type\_task) REFERENCES type\_task(id\_type\_task),

FOREIGN KEY (id\_person\_add) REFERENCES person(id\_person)

)

CREATE TABLE has\_task\_person (

id\_has\_task\_person INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

id\_task INTEGER NOT NULL,

id\_person INTEGER NOT NULL,

FOREIGN KEY (id\_person) REFERENCES person(id\_person),

FOREIGN KEY (id\_task) REFERENCES task(id\_task)

)

CREATE TABLE type\_activity (

Продолжение приложения Г

id\_type\_activity INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

name\_activity TEXT NOT NULL

)

CREATE TABLE log\_time\_task (

id\_log INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

date\_log DATE NOT NULL,

description TEXT ,

links\_ext\_stor TEXT,

id\_type\_activity INTEGER NOT NULL,

id\_has\_task\_person INTEGER NOT NULL,

FOREIGN KEY (id\_type\_activity) REFERENCES type\_activity(id\_type\_activity),

FOREIGN KEY (id\_has\_task\_person) REFERENCES has\_task\_person(id\_has\_task\_person)

)

ALTER TABLE city ADD CONSTRAINT city\_fk0 FOREIGN KEY id\_country REFERENCES country(id\_country)

ALTER TABLE has\_task\_person ADD CONSTRAINT has\_task\_person\_fk0 FOREIGN KEY id\_task REFERENCES taskid\_task

ALTER TABLE has\_task\_person ADD CONSTRAINT has\_task\_person\_fk1 FOREIGN KEY id\_person REFERENCES personid\_person

ALTER TABLE log\_time\_task ADD CONSTRAINT log\_time\_task\_fk0 FOREIGN KEY id\_type\_activity REFERENCES type\_activityid\_type\_activity

ALTER TABLE log\_time\_task ADD CONSTRAINT log\_time\_task\_fk1 FOREIGN KEY id\_has\_task\_person REFERENCES has\_task\_personid\_has\_task\_person

ALTER TABLE person ADD CONSTRAINT person\_fk0 FOREIGN KEY id\_user) REFERENCES userid\_user

ALTER TABLE person ADD CONSTRAINT person\_fk1 FOREIGN KEY id\_post) REFERENCES postid\_post

ALTER TABLE person ADD CONSTRAINT person\_fk2 FOREIGN KEY id\_department) REFERENCES departmentid\_department

ALTER TABLE person ADD CONSTRAINT person\_fk3 FOREIGN KEY id\_city) REFERENCES cityid\_city

ALTER TABLE task ADD CONSTRAINT task\_fk0 FOREIGN KEY id\_type\_task) REFERENCES type\_taskid\_type\_task