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 5](#_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 6](#_Toc469451503)

[2. Анализ процесса управления рабочим временем сотрудников it-компании 7](#_Toc469451504)

[2.1 Описание процесса руководства и проверки курсового проекта 7](#_Toc469451505)

[2.2 Анализ процесса управления рабочим временем сотрудников IT-компании 7](#_Toc469451506)

[3. Разработка автоматизированной системы управления рабочим временем сотрудников it-компании 8](#_Toc469451507)

[3.1 Постановка задачи и описание вариантов использования системы 8](#_Toc469451508)

[3.2 Выбор и обоснование архитектурных решений 8](#_Toc469451509)

[3.3 Функциональное моделирование процесса управления рабочим временем 8](#_Toc469451510)

[3.4 Информационное моделирование системы управления рабочим временем 8](#_Toc469451511)

[3.5 Разработка моделей представления системы управления рабочим временем 8](#_Toc469451512)

[3.6 Описание алгоритмов программных модулей системы управления рабочим временем 8](#_Toc469451513)

[3.7 Результаты проектирования системы проверки курсового проекта 8](#_Toc469451514)

[Заключение 9](#_Toc469451515)

[Список использованных источников 10](#_Toc469451516)

[Приложение А](#_Toc469451517) [(обязательное)](#_Toc469451518) [Функциональная модель «TO BE»  
 (к пункту 3.3) 11](#_Toc469451519)

[Приложение Б](#_Toc469451520) [(обязательное)](#_Toc469451521) [Cхема функции (к пункту 3.6) 12](#_Toc469451522)

[Приложение В](#_Toc469451523) [(обязательное)](#_Toc469451524) [Листинг кода основных элементов   
 программы 13](#_Toc469451525)

[Приложение Г](#_Toc469451526) [(обязательное)](#_Toc469451527) [Листинг скрипта создания базы данных 14](#_Toc469451528)

ВВЕДЕНИЕ

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:

1. Исследовать процесс планирования и учета рабочего времени.
2. Провести анализ существующих аналогов разрабатываемой системы.
3. Провести анализ применяемых инструментальных и технических средств.
4. Охарактеризовать основные процессы планирования и учета рабочего времени.
5. Определить базовые принципы, которыми необходимо руководствоваться при проектировании системы.
6. Сформулировать архитектурное решение для разработки системы.
7. Выбрать и обосновать технические, технологические и программные средства, которые будут использованы при проектировании системы.
8. Выполнить программную реализацию системы.
9. Провести тестирование программного продукта.

# ИССЛЕДОВАНИЕ ОРГАНИЗАЦИИ ПРОЦЕССА УПРАВЛЕНИЯ РАБОЧИМ ВРЕМЕНЕМ

* 1. Сравнительный (критический) анализ существующих аналогов систем управления рабочим временем
  2. Общие понятия и термины

# АНАЛИЗ ПРОЦЕССА УПРАВЛЕНИЯ РАБОЧИМ ВРЕМЕНЕМ СОТРУДНИКОВ IT-КОМПАНИИ

* 1. Описание процесса руководства и проверки курсового проекта
  2. Анализ процесса управления рабочим временем сотрудников IT-компании

# 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.

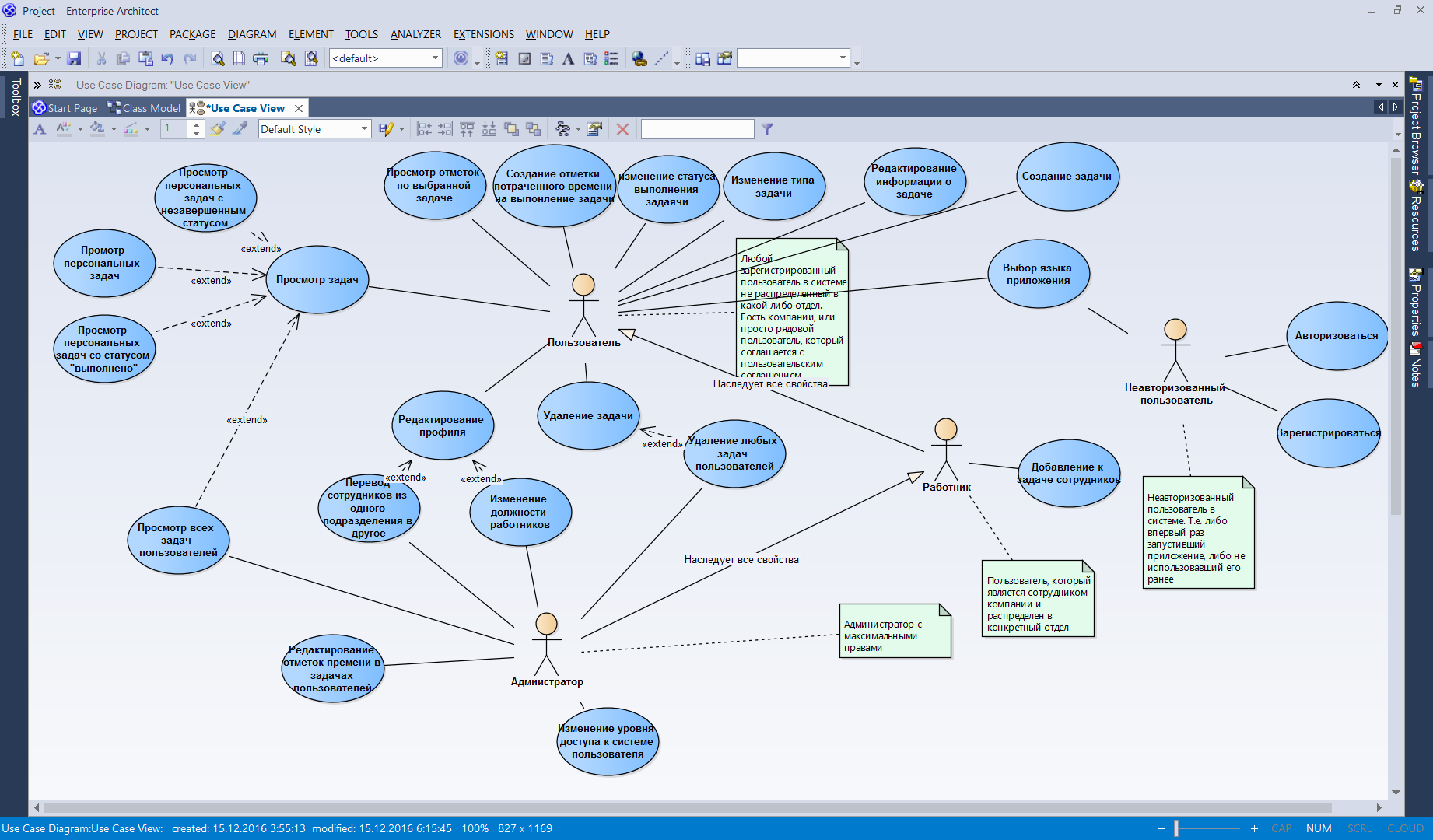


Рисунок 3.1 - Диаграмма вариантов использования системы

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.

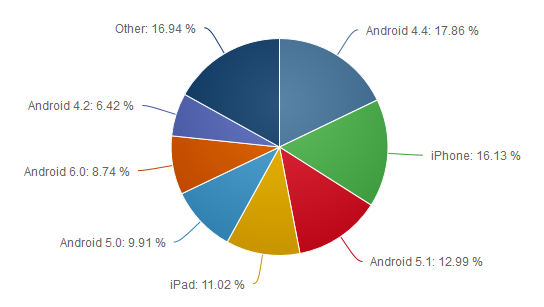


Рисунок 3.2 – процентное соотношение Android к другим опрационным системам

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.

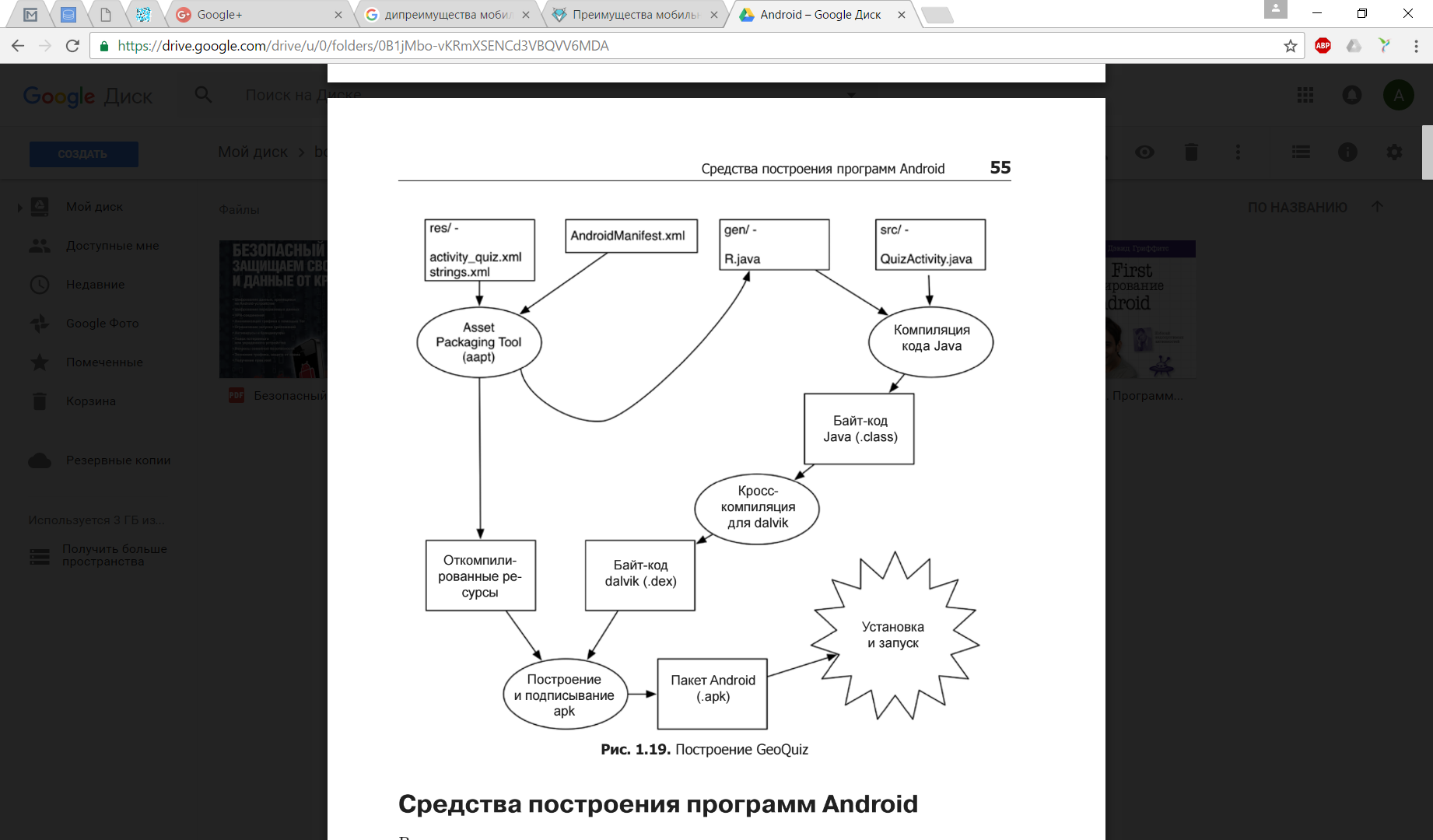


Рисунок 3.2 – Этапы построения Android приложения

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.

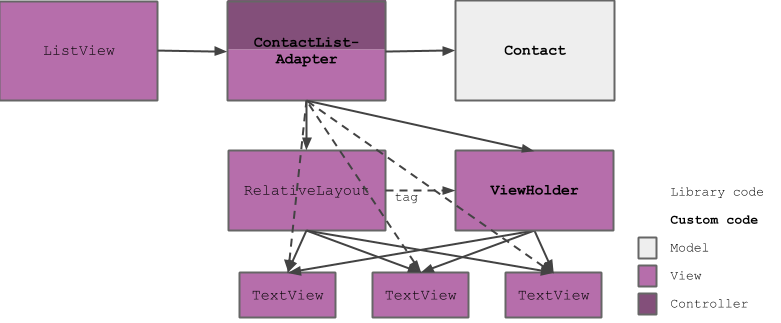


Рисунок 3.3 – Паттерн ViewHolder

[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
  2. 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.

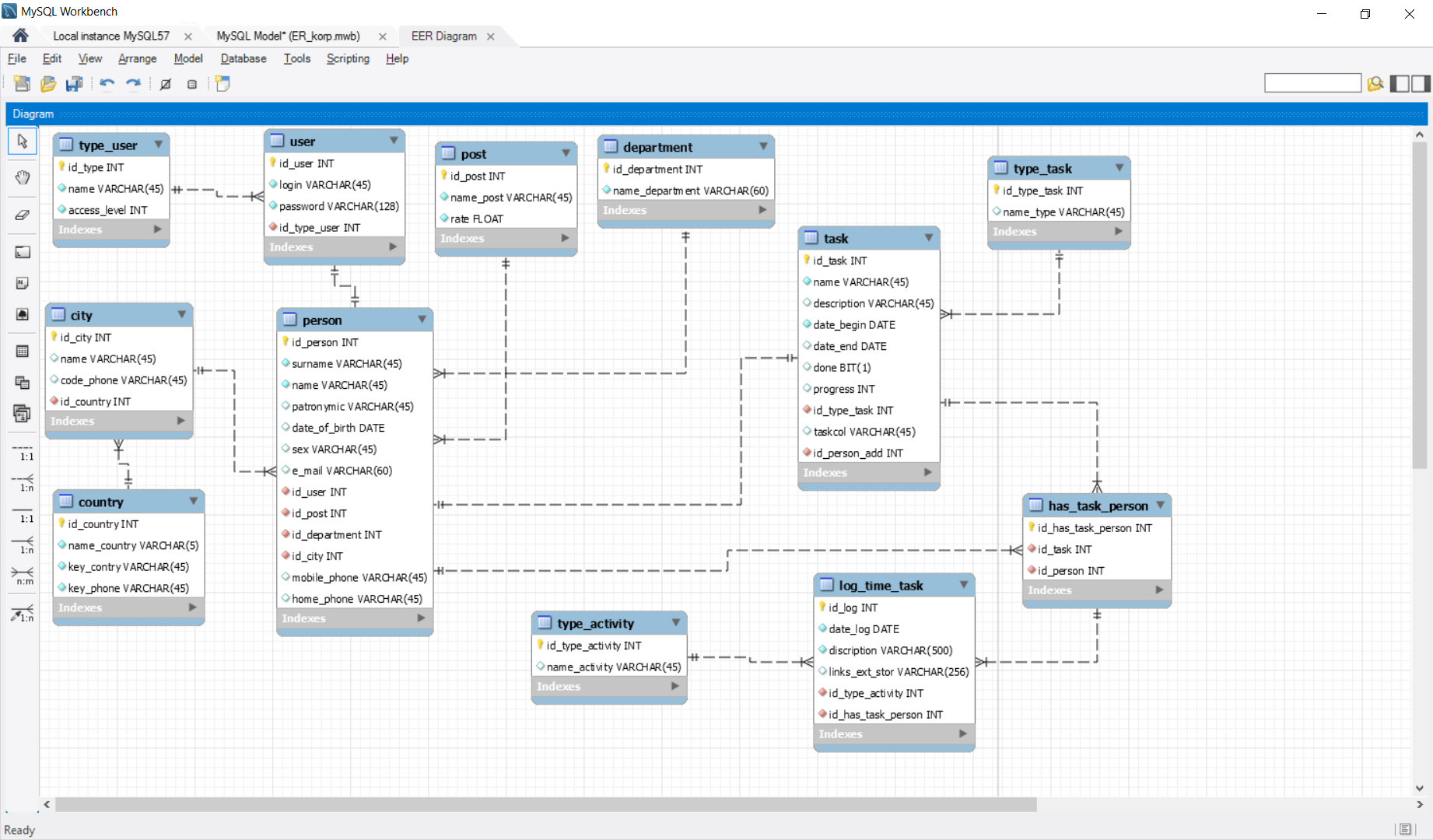


Рисунок 3. – Информационная модель системы управления временем сотрудников ИТ-компании

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
  2. Описание алгоритмов программных модулей системы управления рабочим временем
  3. Результаты проектирования системы управления рабочим временем

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Приложение А

(обязательное)

Функциональная модель «TO BE» (к пункту 3.3)

Приложение Б

(обязательное)

Cхема функции (к пункту 3.6)

Приложение В

(обязательное)

Листинг кода основных элементов программы

Приложение Г

(обязательное)

Листинг скрипта создания базы данных