Министерство образования Республики Беларусь УО «Полоцкий государственный университет»

Кафедра технологий программирования

**О Т Ч Е Т**

о прохождении технологической практики

студента 3 курса 16-ИТ-3 группы Яблонский А.С.

в период с 11.06.2018 по 06.07.2018

|  |  |
| --- | --- |
| Наименование базы  практики: | ЗАО «Кьюликс Системс» |
| Руководитель от предприятия: (должность) | Фурс А. А.  Системный-архитектор |
| Руководитель от кафедры: (ученая степень, звание) | Бураченок И.Б.  ст. преподаватель |

Практика защищена

с оценкой

Дата защиты

Полоцк 2019

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_bookmark0)

1. [ХАРАКТЕРИСТИКИ ПРЕДПРИЯТИЯ](#_bookmark1) 5
   1. [История развития, общие сведения о предприятии 5](#_bookmark2)
2. [НОРМАТИВНАЯ ДОКУМЕНТАЦИЯ ..6](#_bookmark7)
   1. [Содержание должностных инструкций ..6](#_bookmark8)
   2. [Модели жизненного цикла программного обеспечения ..8](#_bookmark9)
3. [ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ИСПОЛЬЗУЕМОЕ В ОРГАНИЗАЦИИ](#_bookmark10)

[.................................................................................................................................11](#_bookmark10)

* 1. [Анализ степени автоматизации в организации 11](#_bookmark11)
  2. [Среды разработки и языки программирования, используемые в](#_bookmark14)

[организации 11](#_bookmark14)

* 1. [Операционные системы, используемые в организации 11](#_bookmark15)
  2. [Требования, предъявляемые к разрабатываемому программному продукту 12](#_bookmark16)

1. [ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ 12](#_bookmark18)
   1. [Ознакомление с задачей 12](#_bookmark19)
   2. [Средства решения задачи 12](#_bookmark21)
   3. [Проектирование программного обеспечения 13](#_bookmark22)
   4. [Результат решение задачи 15](#_bookmark26)

[ЗАКЛЮЧЕНИЕ 16](#_bookmark28)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_bookmark29)

[ПРИЛОЖЕНИЕ А](#_bookmark30) (обязательное)  [18](#_bookmark31)

[ПРИЛОЖЕНИЕ Б………………………………………………………………..21](#_bookmark32)

[ПРИЛОЖЕНИЕ В………………………………………………………………..24](#_bookmark34)

# ВВЕДЕНИЕ

Для студента производственная практика – неотъемлемая часть учебного процесса, помогающая сориентироваться и найти себя в профессии. Для предприятий производственная практика хоть и связана с рядом трудностей, однако, является прекрасной возможностью присмотреть себе персонал заранее и привлечь в компанию ценные молодые кадры.

Производственная практика – это практическая часть образовательного процесса в высшем или среднеспециальном учебном заведении, происходящая в организациях в режиме реальной рабочей деятельности. Практика призвана закрепить полученные теоретические знания и навыки, необходимые для присвоения квалификации и итоговой аттестации учащегося как специалиста. Итоги производственной практики оцениваются согласно нормам, принятым в учебном заведении, и вписываются в учебный процесс.

Производственная практика для студента часто становится отправной точкой его профессиональной карьеры. Наиболее распространенная ошибка учащихся – это формальное отношение к процессу прохождении практики, как к еще одному учебному заданию. Чтобы извлечь из практики максимум пользы, необходимо иметь правильный настрой и понимать, что это уникальная возможность «прощупать почву», находясь при этом еще под крылом своего образовательного учреждения. Значительно сэкономив таким образом время и силы, вы не будете делать лишних движений по окончании вуза и будете точно знать, куда идти дальше.

Практика дает возможность студенту закрепить теоретические знания, применить знания и навыки на практике, сориентироваться в реальном рабочем процессе и увидеть подводные камни выбранной специальности, которые не видны в теории и многое другое.

Во время практики студент должен вести дневник, который подписывается руководителем его практики. По окончании практики ее итоги оцениваются наравне с экзаменами и зачетами и отмечаются в зачетной книжке. Также работу студента оценивает руководство практической базы, на которой он трудился, и выдает характеристику.

В данном отчете предоставлена информация о прохождении производственной практики в ЗАО «Кьюликс Системс» г. Минска.

**1 ХАРАКТЕРИСТИКИ ПРЕДПРИЯТИЯ**

**1.1 История развития, общие сведения о предприятии**

Компания [Qulix Systems](https://www.kv.by/post/1057216-mnenie-kompanii-dolzhny-obuchat-qa-samostoyatelno) была основана в 2000 году. Основной специализацией компании изначально являлись услуги по независимому [тестированию программного обеспечения](https://www.kv.by/post/1049702-doroga-v-it-industriyu-fotoreportazh-s-konferencii-it-entrance-2016), а также консалтингу в области обеспечения качества ИТ-проектов. Но с ростом компании ширился и перечень предоставляемых услуг.

Среди основных направлений деятельности [Qulix Systems](https://www.kv.by/post/1057157-professiya-aytishnika-eto-vozmozhnost-poprobovat-mnozhestvo-drugih-sfer-intervyu-s) можно выделить следующие:

* заказная разработка сложных информационных систем корпоративного уровня
* [разработка мобильных приложений](https://www.kv.by/content/razrabotka-mobilnykh-prilozhenii-dlya-biznesa-vzglyad-kompanii-razrabotchika) под новейшие устройства
* услуги по обеспечению качества ПО
* разработка и внедрение систем дистанционного банковского обслуживания
* графический и веб-дизайн
* разработка веб-сайтов.

На сегодняшний день «Кьюликс» более 14 лет предоставляет услуги по разработке и тестированию ПО, а также консалтингу в сфере информационных технологий на рынках России, Европы и США. Один из фокусов деятельности компании – заказная разработка сложных банковских информационных систем, а также комплексных мобильных приложений, в частности, реализации электронных каналов дистанционного банковского обслуживания (ДБО).

Компания занимается созданием банковских решений с учетом индивидуальных пожеланий заказчика и внедрением в работу, учитывая специфику процессов Банка заказчика.

Сегодня Qulix Systems – это международная компания, поставщик услуг по разработке программного обеспечения, тестированию ПО и консалтингу в сфере ИТ. В год разрабатывает до 160 проектов на рынке B2B и B2C решений, используя самые современные и инновационные подходы.

**2 НОРМАТИВНАЯ ДОКУМЕНТАЦИЯ**

**2.1 Содержание должностных инструкций**

1. ОБЩИЕ ПОЛОЖЕНИЯ

1. Ведущий инженер-программист относится к категории специалистов, принимается на работу и увольняется с работы приказом руководителя.

2. На должность ведущего инженера-программиста назначается лицо, имеющее высшее образование соответствующей квалификации.

3. В своей деятельности ведущий инженер-программист руководствуется:

− нормативными документами по вопросам выполняемой работы;

− уставом организации;

− правилами трудового распорядка;

− приказами и распоряжениями руководителя организации (непосредственного руководителя);

− настоящей должностной инструкцией.

4. Ведущий инженер-программист должен знать:

− руководящие и нормативные материалы, регламентирующие методы разработки алгоритмов, программ и использования вычислительной техники при обработке информации;

− основные принципы структурного и объектно-ориентированного программирования; − виды программного обеспечения;

− стандарты программной документации;

− основные методы, средства и методологии разработки программного обеспечения;

− стандартные алгоритмы;

− языки программирования;

− технологию автоматизированной обработки информации;

− виды и порядок оформления технической документации;

− стандарты качества программного обеспечения;

− основные общепринятые технологии в области разработки программных продуктов;

− технико-эксплуатационные характеристики, конструктивные особенности, назначение и режимы работы оборудования, правила его технической эксплуатации;

− основы организации труда и управления;

− основы законодательства о труде;

− правила и нормы охраны труда и пожарной безопасности.

5. Во время отсутствия ведущего инженера-программиста его обязанности выполняет в установленном порядке назначаемый заместитель, 13 несущий полную ответственность за надлежащее исполнение возложенных на него обязанностей.

2. ДОЛЖНОСТНЫЕ ОБЯЗАННОСТИ

Для выполнения возложенных на него функций программист обязан:

1. На основе анализа математических моделей и алгоритмов решения научных, прикладных экономических и других задач разрабатывать программы, обеспечивающие возможность выполнения средствами вычислительной техники алгоритма и поставленной задачи.

2. Участвовать в выборе языка программирования для описания алгоритмов и структур данных.

3. Разрабатывать технологию решения задачи на всех этапах.

4. Определять информацию, подлежащую обработке средствами вычислительной техники, ее объемы, структуру, макеты и схемы ввода, обработки, хранения и вывода, методы ее контроля.

5. Определять объем и содержание данных контрольных примеров, обеспечивающих наиболее полную проверку соответствия программ их функциональному назначению.

6. Осуществлять запуск отлаженных программ и ввод исходных данных, определяемых условиями поставленных задач.

7. Определять возможность использования готовых программных продуктов.

8. Разрабатывать, отлаживать, анализировать и оптимизировать программный код на основе готовых спецификаций.

9. Интегрировать программные компоненты.

10. Проводить проверку программ на основе логического анализа.

11. Проводить отладку разработанных программ, корректировать их в процессе стабилизации и сопровождения.

12. Выполнять работу по унификации процессов разработки программ.

13. Разрабатывать инструкции по работе с программами, оформлять необходимую техническую документацию.

14. Разрабатывать и внедрять методы автоматизации программирования. 15. Участвовать в сопровождении программного обеспечения.

16. Оказывать помощь специалистам на различных стадиях разработки программного обеспечения при сборе и документировании требований пользователя, в разработке спецификаций.

17. Вести и представлять установленную отчетность.

18. Своевременно и точно исполнять производственные приказы, задания, указания и распоряжения руководителя.

19. Соблюдать установленные на предприятии Правила внутреннего трудового распорядка. 14

20. Оказывать содействие и сотрудничать с нанимателем в деле обеспечения здоровых и безопасных условий труда, немедленно сообщать непосредственному руководителю о каждом случае производственного травматизма и профессионального заболевания, а также о чрезвычайных ситуациях, которые создают угрозу здоровью и жизни для него и окружающих, обнаруженных недостатках и нарушениях охраны труда.

21. Принимать необходимые меры по ограничению развития аварийной ситуации и ее ликвидации.

22. Не разглашать конфиденциальную информацию компании, к которой в т.ч. относятся и любые исходные файлы программных продуктов.

3. ПРАВА

Ведущий инженер-программист имеет право:

1. Знакомиться с проектами решений руководства организации, касающимися его деятельности.

2. Вносить на рассмотрение руководства по совершенствованию информационной системы организации в целом.

3. Получать от руководителей структурных подразделений, специалистов информацию и документы, необходимые для выполнения своих должностных обязанностей.

4. Привлекать специалистов всех структурных подразделений организации для решения возложенных на него обязанностей (если это предусмотрено положениями о структурных подразделениях, если нет - с разрешения руководителя организации).

5. Требовать от руководства организации оказания содействия в исполнении своих должностных обязанностей и прав.

6. Принимать участие в обсуждении вопросов охраны труда, выносимых на рассмотрение собраний (конференций) трудового коллектива (профсоюзной организации).

4. ОЦЕНКА РАБОТЫ И ОТВЕТСТВЕННОСТЬ

Работу ведущего инженера-программиста оценивает непосредственный руководитель (иное должностное лицо). Ведущий инженер-программист несет ответственность:

1. За неисполнение (ненадлежащее исполнение) своих должностных обязанностей, предусмотренных настоящей должностной инструкцией,- в пределах, определенных действующим трудовым законодательством Республики Беларусь.

2. За совершенные в процессе осуществления своей деятельности правонарушения - в пределах, определенных действующим административным, уголовным и гражданским законодательством Республики Беларусь.

3. За причинение материального ущерба - в пределах, определенных действующим трудовым, уголовным и гражданским законодательством Республики Беларусь. 15

4. За несоблюдение правил и норм охраны труда, техники безопасности, производственной санитарии и противопожарной защиты - в соответствии с требованиями нормативных правовых актов Республики Беларусь и локальных актов.

**2.2 Модели жизненного цикла программного обеспечения**

Стандарт ISO/IEC 12207 не предлагает конкретную модель ЖЦ и методы разработки ПО (под моделью ЖЦ понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ. Модель ЖЦ зависит от специфики ИС и специфики условий, в которых последняя создается и функционирует). Его регламенты являются общими для любых моделей ЖЦ, методологий и технологий разработки. Стандарт ISO/IEC 12207 описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы. К настоящему времени наибольшее распространение получили следующие две основные модели ЖЦ:

− каскадная модель (70-85 г.г.);

− спиральная модель (86-90 г.г.).

В изначально существовавших однородных ИС каждое приложение представляло собой единое целое. Для разработки такого типа приложений применялся каскадный способ. Его основной характеристикой является разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем. Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

Положительные стороны применения каскадного подхода заключаются в следующем:

− на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;

− выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Стандарт ISO 12207 — Процессы жизненного цикла программного обеспечения — наиболее полно на уровне международных стандартов отражает жизненный цикл, технологию разработки и обеспечения качества сложных программных средств. Жизненный цикл ПО представлен набором этапов, частных работ и операций в последовательности их выполнения и взаимосвязи, регламентирующих ведения разработки на всех стадиях от подготовки технического задания до завершения испытаний ряда версий и окончания эксплуатации ПО. В ЖЦ включаются описания исходной 16 информации, способов выполнения операций и работ, устанавливаются требования к результатам и правилам их контроля, а также к содержанию технологических и эксплуатационных документов. Определяется организационная структура коллективов, распределение и планирование работ, а также контроль за реализацией ЖЦ ПО.

Стандарт определяет архитектуру, процессы, разделы и подразделы ЖЦ ПО, а также перечень базовых работ и детализирует содержание каждой из них. Архитектура ЖЦ ПО в стандарте базируется на трех крупных компонентах:

− основные процессы жизненного цикла ПО и определяющие работы;

− вспомогательные процессы и работы, поддерживающие жизненный цикл ПО;

− организационные процессы и управление жизненным циклом ПО.

Процессы приобретения и/или подготовки к созданию ПО должны начинаться с инициализации проекта, анализа концепции, анализа рынка продуктов, выработки требований и состава поддерживающих документов, создания предварительного плана проекта. Основные работы по созданию сложного комплекса программ рекомендуется начинать с определения состава сопровождающих документов, выбора средств конфигурационного управления и обеспечения качества, а также выбора методов и средств технологического обеспечения разработки всей информационной системы. Кодирование и тестирование каждого компонента ПО должно быть оформлено совокупностью документов, удостоверяющих соответствие компонента первичной спецификации, содержащих тесты и результаты тестирования.

Рекомендуется разрабатывать план работ, включающий комплексирование компонентов, тестирование по всем разделам требований и показателям качества, а также документирование плана, результатов интеграции, использованных тестов, критериев оценки и полученных результатов. Далее программное обеспечение следует подвергать квалификационному (аттестационному) тестированию по всем разделам требований контракта, при широком варьировании тестов, изменениях значений критериев, а также тестировать полноту и адекватность технологической и пользовательской документации реальному программному продукту. Проверенный таким образом комплекс программ интегрируется в вычислительные средства информационной системы, средства визуализации и телекоммуникации.

Эти работы взаимодействуют с работами, обеспечивающими сопровождение программного обеспечения. Специалисты анализируют сообщения об ошибках и предложения на модификацию программного обеспечения, отбирают их на соответствие требованиям контракта и оценивают целесообразность проведения изменений. Подготовленные 17 изменения тестируются и проверяются по критериям, определенным в документации. Процессы документирования программного обеспечения должны охватывать планирование и обеспечение документирования, рекомендации по стандартизации, проектированию и разработке, а также по производству, конфигурационному управлению и сопровождению комплекта документации на ПО. Для обеспечения гарантий качества следует использовать планирование, методологию, процедуры и стандарты поддержки качества программного обеспечения в соответствии с контрактом с учетом доступных ресурсов. Верификация программного обеспечения должна включать ее организацию, планирование и техническое обеспечение. Удостоверение правильности (аттестация) должна гарантировать полное соответствие программного продукта спецификациям, требованиям и документации на программное обеспечение и возможность его надежного функционирования и безопасного применения пользователем.

**3 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ИСПОЛЬЗУЕМОЕ В ОРГАНИЗАЦИИ**

**3.1 Анализ степени автоматизации в организации**

Программное обеспечение, используемое в подразделении, подразделяется на две группы:

1. Прикладное ПО – совокупность программ, предназначенных для выполнения определенных пользовательских задач и рассчитанных на непосредственное взаимодействие с пользователями;

К группе прикладного ПО относятся:

− профессиональные программные средства (записи входящих и исходящих вызовов служебного телефона 101, расчёта необходимого количества огнетушащих средств и покрытие розлива нефтепродуктов, отправки служебной записки при заступлении караула на дежурство о количестве личного состава и технического вооружения, информационно-справочные системы);

− программные средства общего назначения (текстовые редакторы, графические редакторы, архиваторы, файловые менеджеры, программы сканирования и распознавания текста и т. п.);

− программные средства специального назначения (мультимедиа, гипертекстовые системы, системы мониторинга сети, создания и записи образов, удаленного доступа, в т. ч. терминального, FTP-серверы и FTP-клиенты, система электронной почты и т. п.).

1. Системное ПО – комплекс программ, которые обеспечивают эффективное управление компонентами компьютерной системы. Системное ПО не решает конкретные прикладные задачи, а обеспечивает работу других программ. В настоящем Положении термин «системное ПО» включает совокупность ПО общего назначения, предназначенного для решения часто встречающихся задач обработки информации;

К группе системного ПО относятся:

− операционные системы;

− коммуникационное ПО (ПО, предназначенное для работы с коммуникационным оборудованием (модемами, коммутаторами и т. п.));

− программы создания резервных копий, диагностики, управления памятью и др.

**3.2 Среды разработки и языки программирования, используемые в организации**

Для разработки программного обеспечения в отделе используются такие средства как:

− среды разработки ПО (Android Studio);

− языки программирования (JS, .NET, Java, Swift, Objective-C);

− системы управления базами данных (MS SQL Server).

**3.3 Операционные системы, используемые в организации**

В департаменте используются операционные системы Windows 7, 8.1, 10, Linux Mint.

**3.4 Требования, предъявляемые к разрабатываемому программному продукту**

− правильность - функционирование в соответствии с техническим заданием;

− универсальность - обеспечение правильной работы при любых допустимых данных и защиты от неправильных данных;

− надежность (помехозащищенность) - обеспечение полной повторяемости результатов, т. е. обеспечение их правильности при наличии различного рода сбоев;

− точность результатов - обеспечение погрешности результатов не выше заданной;

− защищенность - обеспечение конфиденциальности информации;

− программная совместимость - возможность совместного функционирования с другим программным обеспечением;

− аппаратная совместимость - возможность совместного функционирования с некоторым оборудованием;

− эффективность - использование минимально возможного количества ресурсов технических средств, например, времени микропроцессора или объема оперативной памяти.

**4 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**4.1 Ознакомление с задачей**

На период прохождения практики была поставлена задча, реализовать приложение для управления задачами. Приложение должно позволять создавать, редактировать и удалять следующие сущности:

1. Проект

* Название
* Сокращенное название

1. Задача

* Название
* Работа (часы)
* Дата начала
* Дата окончания
* Статус (Не начата | В процессе | Завершена | Отложена)

1. Сотрудник

* Фамилия
* Имя
* Должность

Между перечисленными выше объектами необходимо реализовать следующие связи:

* Проект может включать в себя от нуля до множества задач
* Один сотрудник может быть назначен на множество задач
* Дополнительное требование: Одна и та же задача может выполняться множеством сотрудников

**4.2 Средства решения задачи**

Разработка приложения производилась на языке Java с использованием библиотек Android.AppCompact. Для сборки проект использовался Gradle; в качестве программной среды - Android Studio.

**4.3 Проектирование программного обеспечения**

Приложение должно содержать следующие основные элементы:

1. Splash screen
2. Модальная форма ожидания (ответа от сервера)
3. Форма конфигурации

* URL сервера
* Количество записей на странице

1. Главное Меню

* Проекты: Отображается форма “Список проектов”
* Задачи: Отображается форма “Список задач”
* Персоны: Отображается форма “Список сотрудников”

1. Форма “Список проектов”

* Колонки:
  + - 1. Идентификатор
      2. Название
      3. Сокращенное название
* Команды уровня формы
  + - 1. Обновить: Загружаются данные с сервера
      2. Добавить: Отображается форма ввода проекта в режиме добавления
* Команды уровня записи
  + - 1. Просмотреть задачи: Отображается форма «Список задач», в списке отображаются задачи, принадлежащие проекту
      2. Изменить: Отображается форма ввода проекта в режиме редактирования
      3. Удалить: Текущая запись удаляется на сервере, список обновляется

1. Форма “Список задач”

* Колонки:
  + - 1. Идентификатор
      2. Статус в виде иконки (отдельная иконка на каждое состояние)
      3. Проект (Сокращенное название) – не отображается в списке задач, принадлежащих проекту
      4. Название
* Команды уровня формы
  + - 1. Обновить: Загружаются данные с сервера
      2. Добавить: Отображается форма ввода задачи в режиме добавления
* Команды уровня записи
  + - 1. Изменить: Отображается форма ввода задачи в режиме редактирования
      2. Удалить: Текущая запись удаляется на сервере, список обновляется

1. Форма “Список сотрудников”

* Колонки:
  + - 1. Идентификатор
      2. Фамилия
      3. Имя
      4. Должность
* Команды уровня формы
  + - 1. Обновить: Загружаются данные с сервера
      2. Добавить: Отображается форма ввода сотрудника в режиме добавления
* Команды уровня записи
  + - 1. Изменить: Отображается форма ввода сотрудника в режиме редактирования
      2. Удалить: Текущая запись удаляется на сервере, список обновляется

1. Форма ввода проекта

* Поля:
  + - 1. Идентификатор: порядковый номер проекта; формируется автоматически; недоступно для изменения
      2. Название
      3. Сокращенное название
* Команды:
  + - 1. Сохранить: введенные данные отправляются на сервер; управление передается в форму “Список проектов”
      2. Отмена: управление передается в форму “Список проектов”

1. Форма ввода задачи

* Поля:
  + - 1. Идентификатор: порядковый номер задачи; формируется автоматически; недоступно для изменения
      2. Проект: выбирается из списка проектов; если форма открыта из списка задач, принадлежащих проекту, то данное поле установлено равным текущему проекту и недоступно для редактирования
      3. Название
      4. Работа (количество времени необходимого для выполнения задачи, часы)
      5. Дата начала
      6. Дата окончания
      7. Статус (Не начата | В процессе | Завершена | Отложена)
      8. Исполнитель: выбирается из списка персон; (см. [Дополнительные требования](https://wiki.qulix.com/wiki/doku.php?id=training:sandbox:tasks:todo_manager#%D0%B4%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))
* Команды:
  + - 1. Сохранить: введенные данные отправляются на сервер; управление передается в форму “Список задач”
      2. Отмена: управление передается в форму “Список задач”

1. Форма ввода персоны (исполнителя)

* Поля:
  + - 1. Идентификатор; формируется автоматически; недоступно для изменения
      2. Фамилия
      3. Имя
      4. Должность
* Команды:
  + - 1. Сохранить: введенные данные отправляются на сервер; управление передается в форму “Список персон”
      2. Отмена: управление передается в форму “Список персон”

Замечание по терминологии: “команда” обозначает любой элемент управления, используемый для запуска операции, к примеру, это может быть кнопка, пиктограмма, гиперссылка и т.п. Приветствуются любые улучшения в функциональности приложения (к примеру, добавление сортировки или фильтрации, контроль вводимых данных и.т.п.) – если это не идет в разрез с качеством разработки.

**4.4 Результат решение задачи**

В ходе технологической практики было реализовано мобильно приложение «Traning Task», которое соответствует вышеописанным требованиям. Скриншоты разработанного приложения находятся в ПРИЛОЖЕНИИ А. Листинг основных классов приложения находятся в ПРИЛОЖЕНИИ Б, В.

**ЗАКЛЮЧЕНИЕ**

По итогам прохождения производственной практики я получил опыт в разработке программного обеспечения. Мною были получены знания о принципах разработки проекта и изучены технологии, используемые в данной платформе.

Во время работы над практическим заданием был получен опыт работы в организации, изучена документация, изучены технологии для разработки сайтов, применены теоретические знания, полученные в университете.

Были получены знания о организации компании ЗАО «Qulix Systems». Я узнал о их сфере деятельности, предоставляемых услугах, реализуемых проектах. Получил опыт работы в команде.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Мартин Фаулер – Рефакторинг. Улучшение существующего кода – Мн.: Санкт-Петербург, 2013.
2. [Стив МакКоннелл](https://oz.by/people/more907177.html) – Совершенный код. Мастер-класс – СПб: Питер, 2019.
3. Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org. Дата обращения: 05.11.2019.
4. ГОСТ 19.003-80 ЕСПД. Схемы алгоритмов и программ. Обозначения условные графические. – М.: Изд-во стандартов, 1982.

**ПРИЛОЖЕНИЕ А** (обязательное)

**Руководство оператора**

**А1 Назначение приложения**

Мобильное приложение для управления задачами

**А2 Выполнение программы**

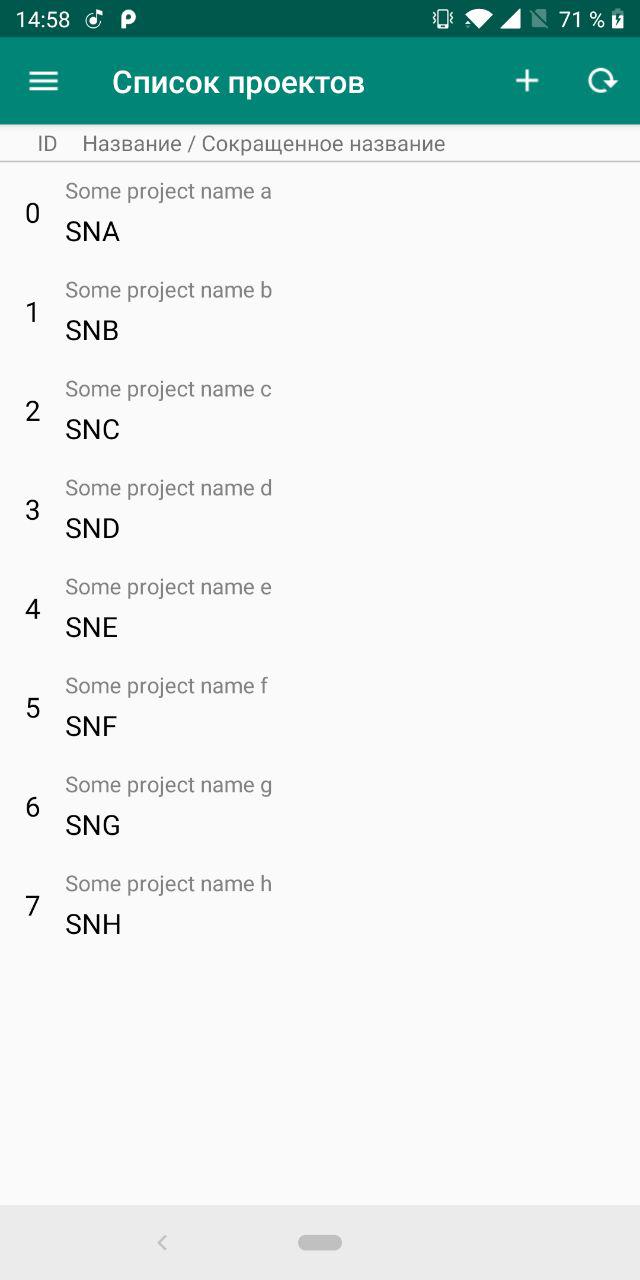


Рисунок 1 – Экран списока проектов

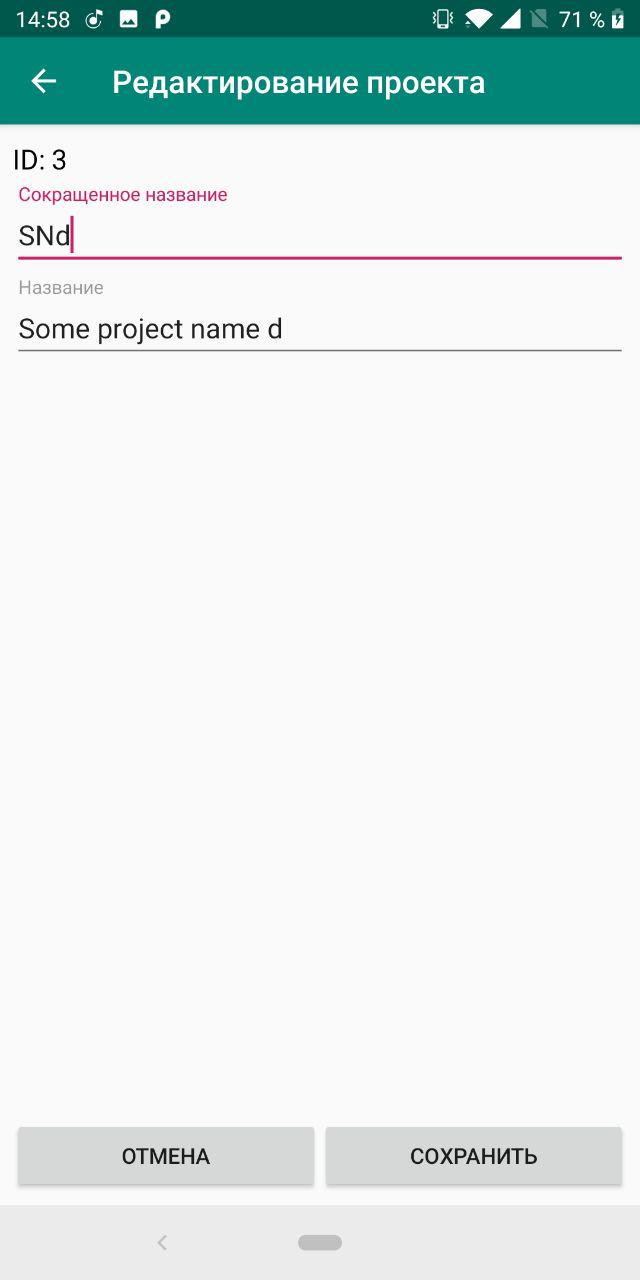


Рисунок 2 – Экран редактирования проекта

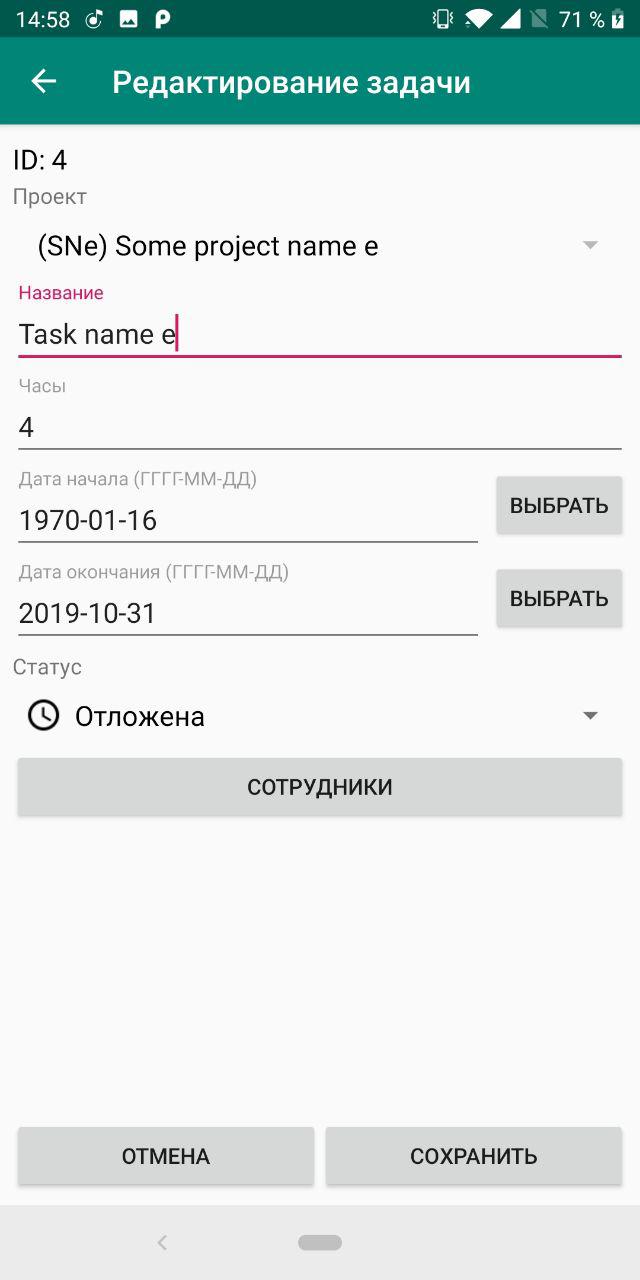


Рисунок 3 – Экран редактирования задачи

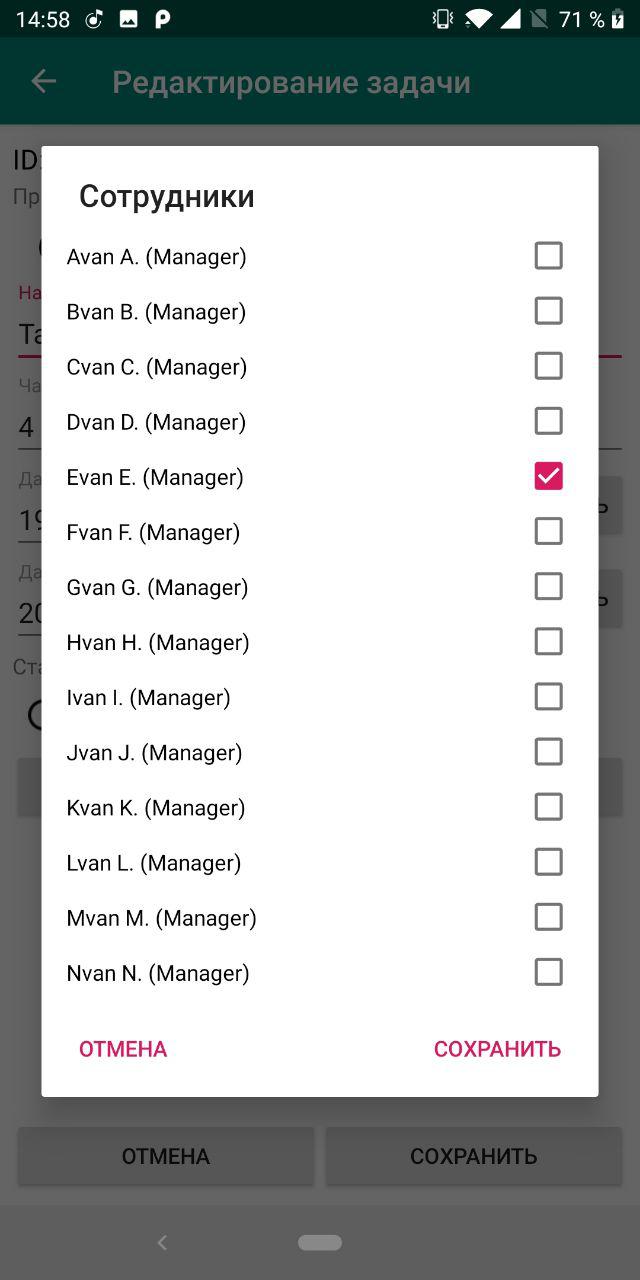


Рисунок 4 – Экран выбора сотрудников на исполнение задачи



Рисунок 5 – Экран настроек

**ПРИЛОЖЕНИЕ Б – ЛИСТИНГ АДАПТЕРА СПИСКА**

1. package com.qulix.yablonskyas.trainingtask.app.utils.adapter;
2. import java.util.ArrayList;
3. import java.util.List;
4. import com.qulix.yablonskyas.trainingtask.app.utils.util.diff.ListDiffUtilCallback;
5. import com.qulix.yablonskyas.trainingtask.app.utils.util.diff.RecyclerAdapterDiffUtil;
6. import android.support.annotation.NonNull;
7. import android.support.v7.widget.RecyclerView;
8. /\*\*
9. Расширяет {@link RecyclerView.Adapter} позволяя устанавливать/изменять/возвращать список элементов.
10. При установки нового списка использует {@link RecyclerAdapterDiffUtil} для просчета изменившихся элементов,
11. реализуя {@link RecyclerAdapterDiffUtil.ListItemsAdapter}.
12. \*
13. @author Andrey Yablonsky
14. \*/
15. public abstract class ModifiableRecyclerAdapter<I, VH extends RecyclerView.ViewHolder>
16. extends RecyclerView.Adapter<VH> implements RecyclerAdapterDiffUtil.ListItemsAdapter<I> {
17. private List<I> items = new ArrayList<>();
18. private ListDiffUtilCallback.ItemsSamePredicate<I> itemsSamePredicate;
19. /\*\*
20. Основной конструктор.
21. \*
22. @param itemsSamePredicate Предикат для сравнения содержимого элементов списка.
23. \*/
24. public ModifiableRecyclerAdapter(ListDiffUtilCallback.ItemsSamePredicate<I> itemsSamePredicate) {
25. this.itemsSamePredicate = itemsSamePredicate;
26. }
27. @Override
28. public List<I> getItems() {
29. return items;
30. }
31. /\*\*
32. Устанавливает новый список элементов
33. \*/
34. public void setItems(List<I> items) {
35. RecyclerAdapterDiffUtil.updateDataList(this, items, itemsSamePredicate);
36. }
37. /\*\*
38. Возвращает элемент, который находит на позиции #position
39. \*/
40. public I getItem(int position) {
41. return items.get(position);
42. }
43. @Override
44. public final int getItemCount() {
45. return items.size();
46. }
47. @Override
48. public final void onBindViewHolder(@NonNull VH holder, int position) {
49. onBindViewHolder(holder, getItem(position));
50. }
51. /\*\*
52. Отображает данные из #item на View из #holder.
53. \*/
54. public abstract void onBindViewHolder(@NonNull VH holder, @NonNull I item);
55. }

**ПРИЛОЖЕНИЕ В – ЛИСТИНГ УТИЛИТНОГО КЛАССА**

1. package com.qulix.yablonskyas.trainingtask.app.utils.util;
2. import java.text.DateFormat;
3. import java.text.ParseException;
4. import java.text.SimpleDateFormat;
5. import java.util.Date;
6. import java.util.Locale;
7. import java.util.Random;
8. import android.graphics.PorterDuff;
9. import android.graphics.drawable.Drawable;
10. import android.support.annotation.ColorInt;
11. import android.support.annotation.NonNull;
12. import android.support.annotation.Nullable;
13. import android.widget.TextView;
14. /\*\*
15. \* Класс с общими утилитами.
16. \*
17. \* @author Andrey Yablonsky
18. \*/
19. public class Utils {
20. private Utils() {
21. }
22. /\*\*
23. \* Проверяет #object на null. Бросает {@link IllegalArgumentException} с текстом #message, если #object = {@code null}.
24. \*
25. \* @param object объект для проверки.
26. \* @param message текст сообщения ошибки.
27. \*/
28. public static void checkNonNull(@Nullable Object object, String message) {
29. if (object == null) {
30. throw new IllegalArgumentException(message);
31. }
32. }
33. /\*\*
34. \* Преобразует #date в строку согласно формату даты в #dateFormat.
35. \*
36. \* @param dateFormat формат даты в строке. Правила оформления доступны в {@link SimpleDateFormat}
37. \* @param date дата для преобразования в строку
38. \* @return дату #date в виде строки
39. \*/
40. public static String dateToString(@NonNull String dateFormat, @NonNull Date date) {
41. return new SimpleDateFormat(dateFormat, Locale.getDefault()).format(date);
42. }
43. /\*\*
44. \* Преобразует строку #dateStr в объект {@link Date} согласно формату #dateFormat
45. \*
46. \* @param dateFormat ожидаемый формат даты #dateStr
47. \* @param dateStr строка с датой
48. \* @return объект даты, соответствующий #dateStr
49. \* @throws ParseException при несоответствии даты #dateStr формату #dateFormat
50. \*/
51. public static Date stringToDate(@NonNull String dateFormat, @NonNull String dateStr) throws ParseException {
52. DateFormat format = new SimpleDateFormat(dateFormat, Locale.getDefault());
53. // отключение "снисходительности" к строке dateStr
54. format.setLenient(false);
55. return format.parse(dateStr);
56. }
57. /\*\*
58. \* Парсит дату из #dateInput {@link TextView} к {@link Date} используя формат #dateFormat.
59. \* При неудачном парсинге вернет сегодняшнюю дату.
60. \*
61. \* @param dateInput {@link TextView}, содержащее текст с датой.
62. \* @param dateFormat формат даты.
63. \* @return объект {@link Date}
64. \*/
65. public static Date getDateFromTextView(TextView dateInput, String dateFormat) {
66. try {
67. String startDateStr = dateInput.getText().toString();
68. return Utils.stringToDate(dateFormat, startDateStr);
69. }
70. catch (ParseException ignore) {
71. // если не удается пропарсить введенную в текстовое поле даты (String -> Date),
72. // то возвращаем сегодняшнюю.
73. return new Date();
74. }
75. }
76. /\*\*
77. \* Создает случайное число из указанного диапазона
78. \*
79. \* @param random экземпляр класса {@link Random}, который будет использоваться для генерации случайного числа.
80. \* @param minValue нижняя граница допустимого диапазона
81. \* @param maxValue верхняя граница допустимого диапазона
82. \* @return случайное число из диапазона от #maxValue до #maxValue
83. \*/
84. public static int randomIntInRange(Random random, int minValue, int maxValue) {
85. return random.nextInt((maxValue - minValue) + 1) + minValue;
86. }
87. /\*\*
88. \* Изменяет цвет #drawable на #color.
89. \*
90. \* @param drawable объект, цвет которого нужно изменить.
91. \* @param color цвет, которым нужно закравить {@link Drawable}
92. \*/
93. public static void changeDrawableColor(Drawable drawable, @ColorInt int color) {
94. drawable.setColorFilter(color, PorterDuff.Mode.MULTIPLY);
95. }
96. }