Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ЭКОНОМИКИ И УПРАВЛЕНИЯ «НИНХ»

Институт Прикладной информатики

Кафедра Прикладных информационных технологий

К ЗАЩИТЕ

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_ (А.Л. Осипов)

"\_\_" июня 2013 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению высшего профессионального образования

010400.62 «Информационные технологии»

Разработка мобильного приложения «Карта развития ребенка» для платформы Android

Исполнитель,

студент гр. 9097 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (А.А. Титов)

Научный руководитель

к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (С.Н. Терещенко)

Нормоконтроль пройден \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (С.Н. Терещенко)

Новосибирск 2013

Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ЭКОНОМИКИ И УПРАВЛЕНИЯ «НИНХ»

Институт Прикладной информатики

Кафедра Прикладных информационных технологий

**ЗАДАНИЕ**

на выпускную квалификационную работу

по направлению высшего профессионального образования

010400.62 «Информационные технологии»

на тему: «Разработка мобильного приложения «Карта развития ребенка» для платформы Android»

студенту гр. 9097 Титову А.А.

Цель работы:

Разработать мобильное приложение, ориентированное на мобильную операционную систему Android.

Задачи работы:

Провести анализ рынка мобильных приложений.

Спроектировать и разработать базу данных.

Спроектировать и разработать программное обеспечение.

План работы и сроки ее выполнения:

Основные структурные элементы: Введение, Анализ требований к ПО, Разработка ПО, Тестирование ПО , Заключение,Список использованных источников, Приложения. Итоговая версия должна быть предоставлена не позднее 17.06.2013.

Научный руководитель

к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (С.Н. Терещенко)

Задание получено,

студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (А.А. Титов)

**ЗАЯВЛЕНИЕ О САМОСТОЯТЕЛЬНОМ ХАРАКТЕРЕ ВЫПОЛНЕННОЙ РАБОТЫ**

Я, Титов Антон Андреевич студент 4 курса

(фамилия, имя, отчество)

специальности (направления бакалаврской или магистерской подготовки, программа)010400.62 «Информационные технологии» заявляю, что в моей выпускной квалификационной работе

(вид работы)

на тему «Разработка мобильного приложения «Карта развития ребенка» для платформы Android»,

представленной в государственную аттестационную комиссиюдля публичной защиты, не содержится элементов плагиата.

Все прямые заимствования из печатных и электронных источников, а также из за­щищенных ранее письменных работ, кандидатских и докторских диссертаций имеют со­ответствующие ссылки.

Дата Подпись

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc365368934)

[1 Анализ требований к программному обеспечению 8](#_Toc365368935)

[1.1 Анализ рынка мобильных приложений 8](#_Toc365368936)

[1.2 Требования к программному обеспечению 11](#_Toc365368937)

[1.3 Средства разработки 18](#_Toc365368938)

[2 Разработка программного обеспечения 20](#_Toc365368939)

[2.1 Разработка базы данных 20](#_Toc365368940)

[2.2 Разработка программного обеспечения 25](#_Toc365368941)

[2.3 Разработка интерфейса 32](#_Toc365368942)

[3 Тестирование программного обеспечения 41](#_Toc365368943)

[3.1 Тестирование 41](#_Toc365368944)

[3.2 Функциональное тестирование 41](#_Toc365368945)

[3.3 Тестирование производительности 44](#_Toc365368946)

[3.4 Тестирование пользовательского интерфейса 48](#_Toc365368947)

[3.5 Перспективы 50](#_Toc365368948)

[ЗАКЛЮЧЕНИЕ 51](#_Toc365368949)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 54](#_Toc365368950)

[ПРИЛОЖЕНИЯ 57](#_Toc365368951)

# ВВЕДЕНИЕ

Начиная с 2000 года, технологии мобильных устройств активно укрепляются в различных сферах человеческой деятельности. Не так давно, мобильный телефон для людей ассоциировался только со средством для использования услуг связи, предоставляемых мобильными операторами. В нынешнее же время дела обстоят иначе. Технический прогресс не стоит на месте и поэтому производители мобильных устройств стремятся вложить в свой продукт поддержку актуальных на данное время инноваций. Как следствие, современные мобильные телефоны имеют более богатый функционал по сравнению со своими предшественниками. Данный функционал позволяет упростить жизнь человеку. Существует много примеров подтверждающих этот факт. Например, не менее 50% различных покупок в США осуществляется с участием мобильного телефона [32], а, например, в крупных городах России внедряется система оплаты проезда в общественном транспорте с помощью мобильных телефонов [35].

Тенденцией последних лет является замена традиционных способов взаимодействия потребителя с окружающим миром на способы, которые могу быть предоставлены мобильными инновациями. В СМИ неоднократно звучали новости о появлении систем мобильных платежей и прочих удобных людям сервисах. Возможно, что в скором времени телефон станет заменой кошелька, видеокамеры, утренней газеты, пропуска на работу. При этом, во многих случаях, при переносе в мобильное устройство, привычная всем услуга становится гораздо более удобной в использовании, нежели устаревший ее аналог. Например, в случае с электронной газетой существует возможность в короткие сроки узнать дополнительную информацию о каком-либо событие или человеке, просто кликнув на ссылку в статье. Также возможность оплаты проезда в общественном транспорте с помощью мобильного телефона позволит экономить время и избегать возникающих в некоторых случаях неудобств.

Таким образом, из вышесказанного можно сделать предположение о том, что в скором будущем огромное количество используемых в повседневной жизни услуг будут реализованы в мобильных устройствах. На основе данного предположения можно заключить, что весьма перспективными становятся любые разработки в области мобильных технологий. Особенно актуальными являются те разработки, которые включают в себя функционал, реализующий каждодневные действия людей в окружающем их мире.

Помимо развития мобильных технологий и инноваций, не менее популярной сферой в области информационных технологий является разработка мобильных приложений. Мобильные приложения могут быть применены как в бизнесе, так и в повседневной жизни людей. Приложения, ориентированные на развитие бизнеса могут сделать его более привлекательным и современным, также они позволяют привлечь новых клиентов и увеличить число продаж. Данные приложения часто являются аналогами или дополнениями корпоративного программного обеспечения. Другой вид мобильных приложений – это приложения, ориентированные на рядового потребителя. Такие программные продукты зачастую относятся к разряду развлекательных и призваны для того, что бы скрасить времяпрепровождение при использование мобильного устройства. Также существуют мобильные приложения на базе оповещения. Данные продукты располагают функционалом, который позволяет пользователю создавать некий набор оповещений или, по простому, «напоминалок».

В данной выпускной квалификационной работе будут рассмотрены процессы проектирования, разработки и тестирования приложения на базе мобильной операционной системы Android. Объектом исследования в данном контексте выступает мобильное приложение для операционной системы Android. Предметом исследования является приложение для управления данными о развитии ребенка для мобильного устройства.

Работа разбита на ряд структурных элементов, указанных ниже.

1. **Введение**. Данный раздел, краткая информация.
2. **Анализ требований к программе**. Анализ рынка, требования к программе, средства разработки.
3. **Разработка программного обеспечения**. Проектирование и разработка компонентов мобильного приложения.
4. **Тестирование программы**. Проведение различных методик тестирования программы.
5. **Заключение**. Выводы по итогам разработки.
6. **Список использованных источников**.
7. **Приложения**.

В качестве средств реализации использованы следующие продукты: интегрированная среда разработки Eclipse, плагин ADT, сервис контроля версий Git, визуальный клиент Tortoise Git, векторный графический редактор Microsoft Visio, редактор ER-моделей DeZign for Databases, плагин ObjectAid Class Diagram, компонент среди разработки DDMS, программа SQLite Database Browser.

# Анализ требований к программному обеспечению

## Анализ рынка мобильных приложений

Рынок мобильных приложений во многом зависит от рынка мобильных устройств, в частности от установленных на них операционных систем. Это объясняется тем, что большее количество приложений являются нативными, то есть ориентированы на определенную операционную систему. Далее на диаграмме представлены компании лидеры по выпуску мобильных устройств (рисунок 1).

Рисунок - Диаграмма "Рынок мобильных устройств"

По данным компании IDC [6] лидирующими в рыночном соотношении мобильными операционными системами являются Android и iOS. Операционная система Windows Phone лишь в квартале 2013 года смогла выйти на третье место. Причиной этому послужило тесное сотрудничество финской компании Nokia с компанией Microsoft, производителем операционной системы Windows Phone [1]. За время этого сотрудничества Nokia выпустила более 20 миллионов устройств с установленной на них операционной системой производства Microsoft. Рыночное соотношение мобильных ОС представлено на следующей диаграмме (рисунок 2):

Рисунок - Диаграмма "Мобильные операционные системы",

При анализе рынка мобильных приложений можно заметить, что этот рынок имеет лидирующие позиции в отношении темпов роста, как отрасль интеллектуальной продукции. Разработка мобильных приложений стала одной из ведущих областей в развитии информационных технологий. В год запуска App Store [2], в 2008 году, рынок мобильных приложений только начинал формироваться, а уже к нынешнему времени он вступил в фазу активного роста. Следствием таких темпов роста является большое внимание к рынку со стороны инвесторов. В денежном выражении, за период 2009-2012 гг., объем рынка мобильных приложений в мире увеличился в 25 раз. Согласно данным J'son & Partners Consulting, за 2012 год рынок мобильных приложений в мире составил 7,83 миллиардов долларов. И, по прогнозам, к 2016 году составит 65,79 миллиардов долларов [20]. Относительно рынка мобильных приложений в России известно следующее: общий объем рынка на 2012 год составил около 160 миллионов долларов, что в 3.5 раза больше, чем в 2011 году. По оценкам J'son & Partners Consulting, российский рынок мобильных приложений к 2016 году может вырасти еще в 8 раз, что составит около 1,3 миллиарда долларов.

Большой вклад в развитие рынка мобильных приложений вносят площадки для распространения и продажи самих приложений. Как правило, в роли этих площадок выступают интернет – магазины мобильных приложений. Во всех случаях данные интернет – магазины ориентированы под одну определенную мобильную операционную систему. Наиболее популярными интернет магазинами являются следующие:

* Apple App Store,
* Android Market [4],
* Microsoft Marketplace [16],
* BlackBerry App World [30].

В следующей таблице (таблица 1) представлена статистика по вышеуказанным интернет – магазинам мобильных приложений.

Таблица - Статистика интернет - магазинов мобильных приложений

|  |  |  |  |
| --- | --- | --- | --- |
| Название интернет магазина | Поддерживаемая платформа | Количество доступных приложений | Общее количество скачанных приложений, млн |
| Apple App Store | iOS | 650000 | 30000 |
| Android Market | Android | 500000 | 15000 |
| Microsoft Marketplace | Windows Phone | 125000 | Данные не разглашаются |
| BlackBerry App World | Blackberry OS | 70000 | 3000 |

По мнению экспертов из компании J’son & Partners Consulting [7] рынок мобильных приложений можно разделить на следующий сегменты:

* контентные приложения – это приложения, позволяющие взаимодействовать с контентом. Например, программы для прослушивания музыки, чтения книг, просмотра фильмов и видео роликов.
* Бизнес приложения – зачастую это приложения, дополняющие корпоративное программное обеспечение. Такие программы позволяют повысить скорость и удобство работников фирмы.
* Мобильные игры – наиболее востребованный сегмент рынка мобильных приложений. Такие приложения используются пользователями того, чтобы скрасить свой досуг или же просто скоротать время.
* Мобильные социальные сети – на мой взгляд, данный сегмент рынка можно поделить на три типа:
  + мобильные версии сайтов интернет сообществ;
  + социальные сети, изначально ориентированные на мобильные устройства;
  + приложения, адаптирующие некий набор функции социальной сети к возможностям мобильного устройства.

## Требования к программному обеспечению

Для предъявления списка требований к разрабатываемой программе было оформлено следующее техническое задание:

1. Цель проекта: разработать программу для хранения данных о ребенке с возможностью добавления заметок и уведомлений на определенную дату.
2. Для кого разрабатывается: приложение разрабатывается для родителей, желающих сохранить памятные моменты своих детей в электронном виде.
3. Основные функции:

* регистрация пользователя;
* авторизация пользователя;
* добавление и редактирование профиля ребенка;
* добавления и изменения изображения в профиле ребенка, идентифицирующего ребенка;
* добавление и редактирование заметок на определенную дату;
* возможность добавления и изменения изображения при редактировании заметок;
* в качестве источников добавляемых изображений необходимо использовать камеру мобильного устройства либо галерею;
* добавление и редактирование уведомлений.

1. Системные требования:

* минимальная версия Android: 2.3.3 (GINGERBREAD\_MR1, API level: 10);
* целевая версия Android: 4.2 (JELLY\_BEAN\_MR1, API level: 17);
* наличие в устройстве камеры.

1. Требования к отображению данных:

* форматы дат должны соответствовать следующим:
  + на формах просмотра и редактирования заметок, редактирования и добавления профиля ребенка: "dd.MM.yyyy" (пример: 01.01.2001);
  + на формах просмотра и редактирования уведомлений: "HH:mm:ss dd-MM-yyyy" (пример: 14:00:00 01.01.2001);
* формат дробных чисел должен ограничиваться тремя цифрами после запятой;
* ограничение строковых полей происходит методом урезания строки без добавления других символов.

1. Требования к формам и вводимым значениям:

* форма аутентификации:
  + значение «имя учетной записи» не должно быть меньше четырех символов;
  + значение «пароль» не должно быть меньше шести символов;
  + в случае введения некорректных значений необходимо вывести сообщение с помощью метода Toast.makeText(…) или с помощью метода текстового поля .setError(…).
* Форма регистрации:
  + значение «имя учетной записи» не должно быть меньше четырех символов;
  + значение «пароль» не должно быть меньше шести символов;
  + значение «подтверждение пароля» должно в точности соответствовать значению «пароль»;
  + в случае соответствия значений «пароль» и «подтверждение пароля» задний фон поля значения «подтверждение пароля» должен быть зеленым, в противном случае красным;
  + поля «имя», «фамилия», «отчество» не должны быть пустыми при выполнении регистрации;
  + в случае введения некорректных значений необходимо вывести сообщение с помощью метода Toast.makeText(…) или с помощью метода текстового поля .setError(…).
* Форма редактирования профиля ребенка:
  + поля «имя», «фамилия», «отчество» не должны быть пустыми при окончании редактирования профиля;
  + поля «вес» и «рост» - дробное или целое число;
  + значение «дата рождения» должно быть оформлено с помощью элемента управления DatePicker;
  + в случае введения некорректных значений необходимо вывести сообщение с помощью метода Toast.makeText(…) или с помощью метода текстового поля .setError(…).
* Форма выбора текущего профиля ребенка:
  + доступные профили должны быть оформлены в виде списка;
  + текущий выбранный профиль должен быть подсвечен зеленым цветом;
  + переход на другие формы осуществляется с помощью элемента управления ActionBar.
* Форма редактирования заметки:
  + поля «заголовок» и «описание» не должны быть пустыми;
  + значение «дата заметки» должно быть оформлено с помощью элемента управления DatePicker;
  + в случае введения некорректных значений необходимо вывести сообщение с помощью метода Toast.makeText(…) или с помощью метода текстового поля .setError(…).
* Форма просмотра заметок на дату:
  + заметки должны отображаться списком в соответствии с текущей выбранной датой;
  + переход на другие формы осуществляется с помощью элемента управления ActionBar.
* Форма редактирования уведомления:
  + поля «заголовок» и «описание» не должны быть пустыми;
  + значение «время уведомление» должно быть оформлено с помощью элемента управления TimePicker;
  + значение «дата уведомления» должно быть оформлено с помощью элемента управления DatePicker;
  + в случае введения некорректных значений необходимо вывести сообщение с помощью метода Toast.makeText(…) или с помощью метода текстового поля .setError(…).
* Форма просмотра уведомлений:
  + уведомления должны отображаться списком в соответствии с текущей выбранной датой;
  + переход на другие формы осуществляется с помощью элемента управления ActionBar.

1. Макеты форм:
   * форма аутентификации (рисунок 3):

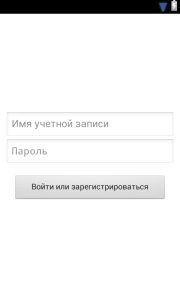


Рисунок – Макет формы аутентификации

* форма регистрации (рисунок 4):

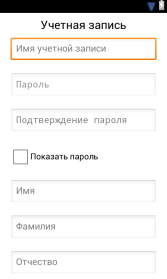


Рисунок – Макет формы регистрации

* форма редактирования профиля ребенка (рисунок 5):

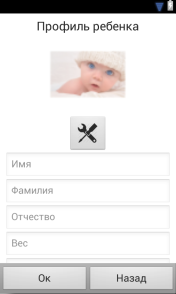


Рисунок – Макет формы редактирования профиля ребенка

* форма выбора текущего профиля ребенка (рисунок 6):

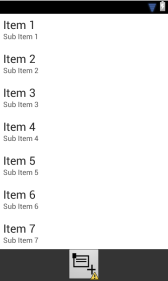


Рисунок – Макет формы выбора текущего профиля ребенка

* форма редактирования заметок (рисунок 7):

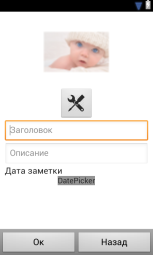


Рисунок – Макет формы редактирования заметок

* форма просмотра заметок на дату (рисунок 8):

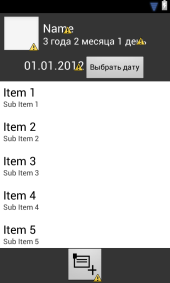


Рисунок – Макет формы просмотра заметок

* форма просмотра уведомлений (рисунок 9):

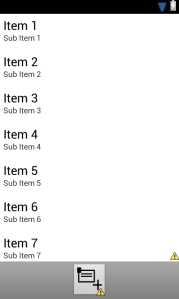


Рисунок – Макет формы просмотра уведомлений

* форма редактирования уведомлений (рисунок 10):

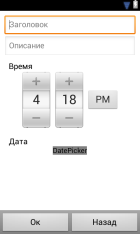


Рисунок - Макет формы редактирования уведомлений

1. Варианты переходов:

* форма аутентификации:
  + форма регистрации;
  + форма выбора профиля ребенка.
* Форма регистрации:
  + форма редактирования профиля ребенка.
* Форма редактирования профиля ребенка:
  + форма выбора профиля ребенка.
* Форма выбора профиля ребенка:
  + форма просмотра заметок;
  + форма редактирования профиля ребенка;
  + форма просмотра уведомлений;
  + форма аутентификации.
* Форма просмотра заметок:
  + форма редактирования заметки;
  + форма выбора профиля ребенка;
  + форма просмотра уведомлений;
  + форма аутентификации.
* Форма редактирования заметки:
  + форма просмотра заметок.
* Форма редактирования уведомления:
  + форма просмотра уведомлений.
* Форма просмотра уведомлений:
  + форма редактирования уведомления;
  + форма просмотра заметок;
  + форма выбора профиля ребенка;
  + форма аутентификации.

1. Поведение приложения при исключительных ситуациях: при возникновении исключительных ситуаций, таких как инициализация исключения приложением, программа должна выполнить откат неудачного действия, если оно было выполнено частично, и выполнить процедуру выхода из учетной записи.

## Средства разработки

В качестве основного средства разработки была выбрана свободная интегрированная среда разработки модульных кроссплатформенных приложений Eclipse (версия 3.8.0) [3]. Данная среда разработки, изначально разрабатываемая фирмой IBM [5], а в дальнейшем переданная независимому от IBM сообществу, широко применяется в области разработки всевозможных программ и приложений. Eclipse имеет полностью открытый исходный код, а архитектура этой среды разработки позволяет дополнять ее различными модулями [24]. В основе IDE Eclipse лежит язык программирования Java, поэтому она является платформо-независимым продуктом [10].

Для разработки приложения под мобильную операционную систему Android был установлен плагин Android Development Tools (ADT), разработанный специально для IDE Eclipse и содержащий в своей основе инструментарий для эмуляции мобильного устройства [14]. Разработка логической части приложения осуществлялась на объектно-ориентированном языке Java [8]. Вёрстка форм приложения выполнена с использованием языка расширяемой разметки XML.

Для хранения данных в мобильном приложении была использована легковесная встраиваемая система управления базами данных Sqlite. Даная СУБД по умолчанию встроена в операционную систему Android [27]. Она поддерживает стандартные возможности реляционных баз данных и, кроме того, требует очень небольшого количество памяти для работы – около 250 килобайт.

В качестве второстепенных средств разработки были выбраны следующие продукты:

* Git – распределенная система контроля версий.
* Tortoise Git – графический интерфейс для контроля версий в репозитории Git.
* Microsoft Visio – векторный графический редактор диаграмм и блок-схем.
* DeZign for Databases – графический редактор ER-моделей.
* ObjectAid Class Diagram – плагин для eclipse, позволяющие строить диаграммы классов.
* SQLite Database Browser – программа для управления структурой и данными базы данных SQlite.
* DDMS (Dalvik Debug Monitor Server) – компонент среды разработки Eclipse, используемый для диагностики разрабатываемого мобильного приложения.

# Разработка программного обеспечения

## Разработка базы данных

Для создания базы данных в первую очередь необходимо выявить объекты, которые в последствии войдут в основу реализации таблиц [15]. При проектировании базы данных разрабатываемого мобильного приложения были определены следующие объекты:

* учетная запись;
* ребенок;
* заметка;
* уведомление.

Определение данных объектов исходит из требований к программе. Также на основании требований были определены атрибуты вышеуказанных объектов:

* учетная запись:
  + имя родителя;
  + фамилия родителя;
  + отчество родителя;
  + дата рождения родителя;
  + имя учетной записи;
  + пароль;
* ребенок:
  + имя;
  + фамилия;
  + отчество;
  + вес;
  + рост;
  + дата рождения;
* заметка:
  + заголовок;
  + описание;
  + дата заметки;
* уведомление:
  + заголовок;
  + описание;
  + дата уведомления.

В основе проектирования базы данных лежит создание логической модели [12]. Она служит для отображения связей таблиц базы данных. Для того, чтобы определить логическую модель необходимо составить перечень полей таблицы и представить описание каждому атрибуту таблицы. При определении наименований полей таблиц проектируемой базы данных использовался английский перевод ранее определенных названий объектов и их атрибутов. При формулировании описания каждого из полей были взяты во внимание предназначение поля и его принадлежность к таблице.

Таблица «Account» содержит сведения о родителе, а также логин и пароль учетной записи. Каждая строка данной таблицы представляет собой экземпляр отдельно взятой учетной записи. Описание полей таблицы представлено ниже (таблица 2).

Таблица – Описание полей таблицы "Account"

| ***Поле*** | ***Описание*** |
| --- | --- |
| AccountId | Идентификационный номер учетной записи, первичный ключ |
| ParentName | Имя родителя |
| ParentSurname | Фамилия родителя |
| ParentMiddlename | Отчество родителя |
| Login | Имя учетной записи |
| Password | Пароль от учетной записи |

Таблица «Children» содержит сведения о ребенке. Описание полей таблицы представлено ниже (таблица 3).

Таблица - Описание полей таблицы "Children"

| ***Поле*** | ***Описание*** |
| --- | --- |
| ChildrenId | Идентификационный номер ребенка, первичный ключ |
| Weight | Весь ребенка |
| Growth | Рост ребенка |
| ChildrenName | Имя ребенка |
| ChildrenSurname | Фамилия ребенка |
| ChildrenMiddlename | Отчество ребенка |
| AccountId | Идентификационный номер учетной записи, внешний ключ |
| Awatar | Фотография ребенка |

Таблица «Note» содержит сведения о добавляемых заметках. Описание полей таблицы представлено ниже (таблица 4).

Таблица – Описание полей таблицы "Note"

| ***Поле*** | ***Описание*** |
| --- | --- |
| NoteId | Идентификационный номер заметки, первичный ключ |
| Description | Описание заметки |
| Postdate | Дата заметки |
| ChildrenId | Идентификационный номер ребенка, внешний ключ |
| Photo | Изображение, хранящееся в заметке |
| Title | Заголовок заметки |

Таблица «Notice» содержит сведения о добавляемых уведомлениях. Описание полей таблицы представлено ниже (таблица 5).

Таблица –Описание полей таблицы "Notice"

| ***Поле*** | ***Описание*** |
| --- | --- |
| NoticeId | Идентификационный номер уведомления, первичный ключ |
| NotifiDateTime | Дата и время уведомления |
| Title | Заголовок уведомления |
| Description | Описание уведомления |
| ChildrenId | Идентификационный номер ребенка, внешний ключ |

На рисунке (рисунок 11) изображена логическая модель базы данных:



Рисунок - Логическая модель базы данных

Для представления типов данных в таблицах реализована физическая модель (рисунок 12).

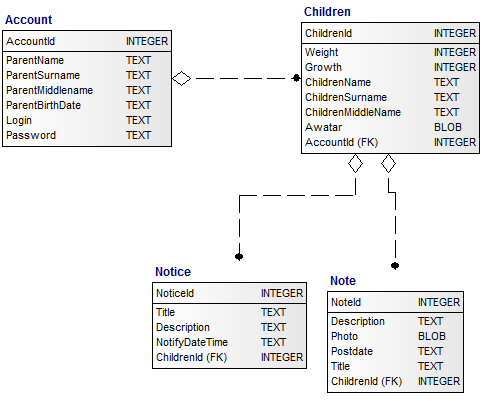


Рисунок - Физическая модель базы данных

На основе логической и физической моделей были написаны скрипты для реализации базы данных в мобильном приложении. Для отображения базы реализации базы данных использована программа SQLite DataBase Browser. Данная программа представляет собой редактор файлов баз данных СУБД SQLite. Она позволяет изменять структуру базы данных, изменять набор полей выбранной таблицы и просматривать и изменять строки таблиц. Реализация базы данных в программе представлена ниже (рисунок 13).

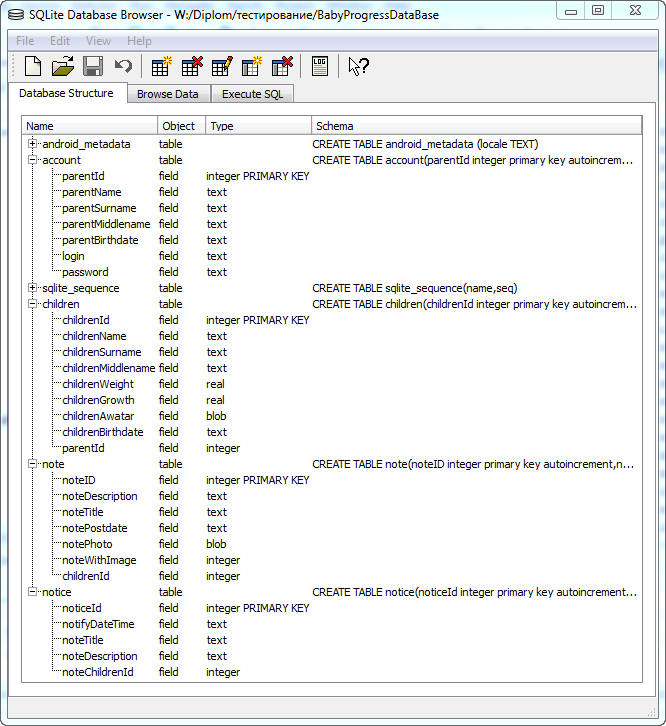


Рисунок - Реализация базы данных в программе

Из рисунка (рисунок 13) видно, что все таблицы базы данных созданы в соответствии с логической и физической моделями. Помимо проектируемых таблиц, операционной системой Android были созданы две служебные таблицы:

* android\_metadata – таблица, содержащая информацию о локализации;
* sqlite\_sequence – таблица, содержащая информацию о количестве строк всех таблиц базы данных.

## Разработка программного обеспечения

Для проектирования программного обеспечения был использован графический язык UML. UML – язык графического описания объектного моделирования в области разработки программного обеспечения. В процессе проектирования были созданы следующие диаграммы:

* диаграмма прецедентов;
* диаграмма классов, взаимодействующих с базой данных;
* диаграмма классов без непосредственного взаимодействия с базой данных;
* диаграмма классов, определяющих паттерн «состояние».

Диаграмма прецедентов – это диаграмма, отражающая отношения различных актеров с имеющимися прецедентами. Под актерами понимается некая роль, взаимодействующая с рассматриваемым объектом, а под прецедентом – некая последовательность действий. Эта диаграмма позволяет описать систему на концептуальном уровне. В данном случае были выявлены следующие актеры и соответствующие им прецеденты (таблица 6):

Таблица - Актеры и соответствующие им прецеденты

|  |  |
| --- | --- |
| **Прецедент** | **Актер** |
| регистрация | пользователь |
| авторизация | пользователь |
| редактирование профиля ребенка | пользователь |
| редактирование заметки | пользователь |
| редактирование уведомления | пользователь |
| инициализация уведомления | мобильное устройство |

Таким образом, исходя из данных таблицы, можно построить следующую диаграмму прецедентов (рисунок 14):



Рисунок - Диаграмма прецедентов

Для реализации объектно-ориентированных методик разработки программного обеспечения необходимо спроектировать классы программы [19]. В первую очередь, для проектирования классов, за основу были взяты объекты базы данных. Эти классы необходимы для использования данных, хранящихся в базе данных. Использование подразумевает собой добавление данных в базу, редактирование и удаление [22]. Также были спроектированы классы, используемые в кастомизации стандартных элементов управления, формировании логики программы и выполнении процедур по конвертации данных [17]. В качестве процедур по конвертации данных были использованы следующие процедуры:

* преобразование дат к необходимым форматам;
* преобразование используемых пользователем изображений в массивы байт;
* преобразование массивов байт в изображение.

Перечень спроектированных классов:

* Account – физическая реализация учетной записи.
* Children – физическая реализация профиля ребенка.
* Note – физическая реализация заметки.
* Notice – физическая реализация уведомления.
* DataManager – класс, предоставляющий методы взаимодействия со статическими переменными проекта, методы для конвертации данных.
* DataAdapter – класс, предоставляющий методы взаимодействия с базой и статические переменные форматов дат.
* ActivityEnum – перечисляемый тип данных, множеством значений которого являются идентификаторы форм.
* MyAsyncTask – класс, необходимый для запуска асинхронного запроса с выводом сообщения на экран.
* ListViewNoteAdapter – адаптер кастомизированных пунктов элемента управления ListView, в котором содержится список заметок.
* ListViewChildrenAdapter – адаптер кастомизированных пунктов элемента управления ListView, в котором содержится список профилей детей.
* ListViewNoticeAdapter – адаптер кастомизированных пунктов элемента управления ListView, в котором содержится список уведомлений.
* BabyProgressDataBaseHelper – класс, содержащий скрипты создания таблиц базы данных и статические константы, необходимые для взаимодействия с базой данных.
* State – абстрактный класс, от которого происходит наследование классов – состояний.
* ChangeChildrenProfileState – класс, реализующий состояние пребывания на форме выбора профиля ребенка.
* ChildrenProfileState – класс, реализующий состояние пребывания на форме профиля ребенка.
* EditNoteState – класс, реализующий состояние пребывания на форме редактирования заметки.
* LoginState – класс, реализующий состояние пребывания на форме аутентификации.
* NotesState – класс, реализующий состояние пребывания на форме просмотра заметок на дату.
* RegistrationState – класс, реализующий состояние пребывания на форме регистрации.
* ViewChildrenProfileState – класс, реализующий состояние пребывания на форме просмотра профиля ребенка.
* ViewNoteState – класс, реализующий состояние пребывания на форме просмотра заметки.
* EditNoticeState – класс, реализующий состояние пребывания на форме редактирования уведомления.
* NotificationsState – класс, реализующий состояние пребывания на форме просмотра заявок.
* NotificationUtils – класс, позволяющий создавать уведомления.
* AlarmReceiver – класс, инициирующий создание уведомления в статус баре на момент наступления даты и времени уведомления.
* NotificationAsyncTask – класс, использующийся для создания уведомления в дополнительном потоке.

При проектировании программного обеспечения были использованы шаблоны проектирования. Шаблон проектирования (паттерн) – повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста. Использованные в работе шаблоны проектирования представлены ниже.

**Одиночка** – шаблон проектирования, гарантирующий, что какой-либо класс имеет только единственный экземпляр, и предоставляющий глобальную точку доступа к этому экземпляру [13].

**Состояние** – шаблон проектирования, который представляет собой реализацию конечного автомата. Задача паттерна состоит в изменении поведения объекта при изменении его внутреннего состояния [27].

**Стратегия** – поведенческий шаблон проектирования, предназначенный для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости. Данный паттерн позволяет менять выбранный алгоритм независимо от объектов – клиентов, которые его используют [9].

**MVC** – схема использования нескольких шаблонов проектирования, с помощью которых модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на остальные [11].

Ниже представлена диаграмма, отображающая связи классов, определяющих шаблон проектирования состояние (рисунок 15).

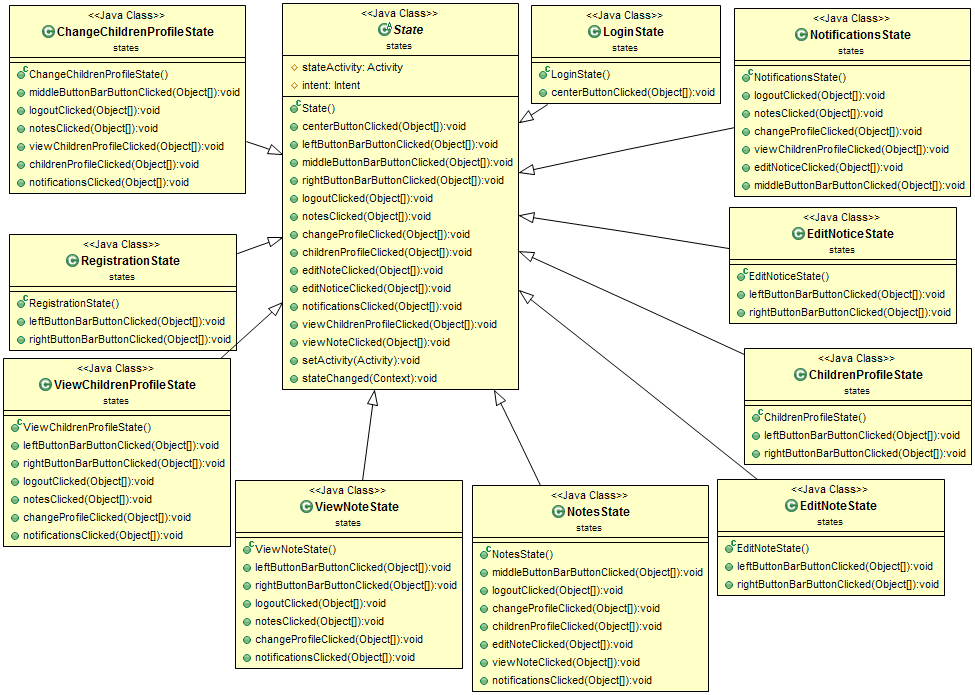


Рисунок - Диаграмма классов, определяющих шаблон проектирования «состояние».

Такой стрелкой, как на диаграмме выше, отображается связь – наследование. На диаграмме видно, что в основе реализованного паттерна «состояние» лежит абстрактный класс State. Этот класс содержит в себе полный перечень событий, которые могу произойти на каком-либо состоянии [26]. Остальные классы на диаграмме являются потомками класса State и частично реализуют его методы. Каждый из классов-потомков представляет собой состояние пребывания на какой-либо форме приложения. Ниже расположена диаграмма классов, которые непосредственно взаимодействуют с локальной базой данных приложения (рисунок 16).

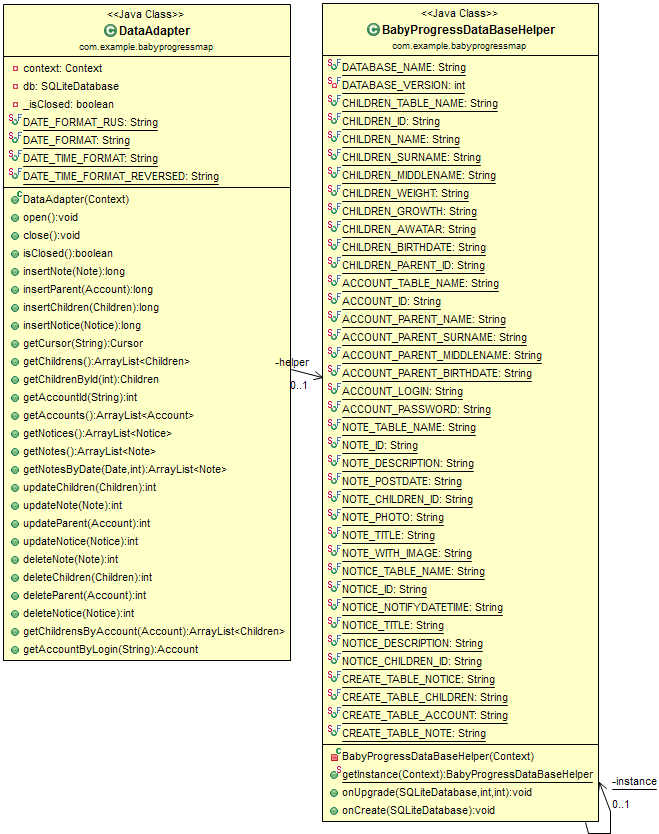


Рисунок - Диаграмма классов, взаимодействующих с базой данных

Класс BabyProgressDataBaseHelper содержит в себе набор статических констант, необходимых для формирования скриптов создания и использования базы данных. На примере данного класса был реализован шаблон проектирования «одиночка», с помощью которого в приложения может существовать только один экземпляр данного класса [23].

Основным назначением класса DataAdapter является предоставление функций по взаимодействию с базой данных, а также статических переменных, которые содержат форматы дат, используемые в приложении. Для взаимодействия с базой данных класс содержит методы для открытия и закрытия подключения к базе [29].

При определении логики программы был спроектирован класс DataManager. Он занимает главенствующую позицию в определении взаимосвязи остальных классов. Данный класс содержит набор статических переменных, необходимых для отслеживания текущего потока данных. Помимо этого, класс DataManager предоставляет методы для осуществления переходов между состояниями приложения. Для реализации переходов по состояниям класс хранит статические переменные, определяющие все возможные состояния и экземпляр класса State, представляющий собой текущее состояние. Кроме этого, DataManager реализует процедуры по конвертации данных и процедуру приведения статических данных в начальное состояние.

При проектировании классов программы также необходимо обозначить классы, которые, в последующем, станут реализацией объектов, хранящихся в базе данных. Таковыми являются следующие классы: Account, Note, Children, Notice.

На следующей диаграмме можно увидеть взаимосвязь классов, которые не имеют непосредственного взаимодействия с базой данных (рисунок 17).

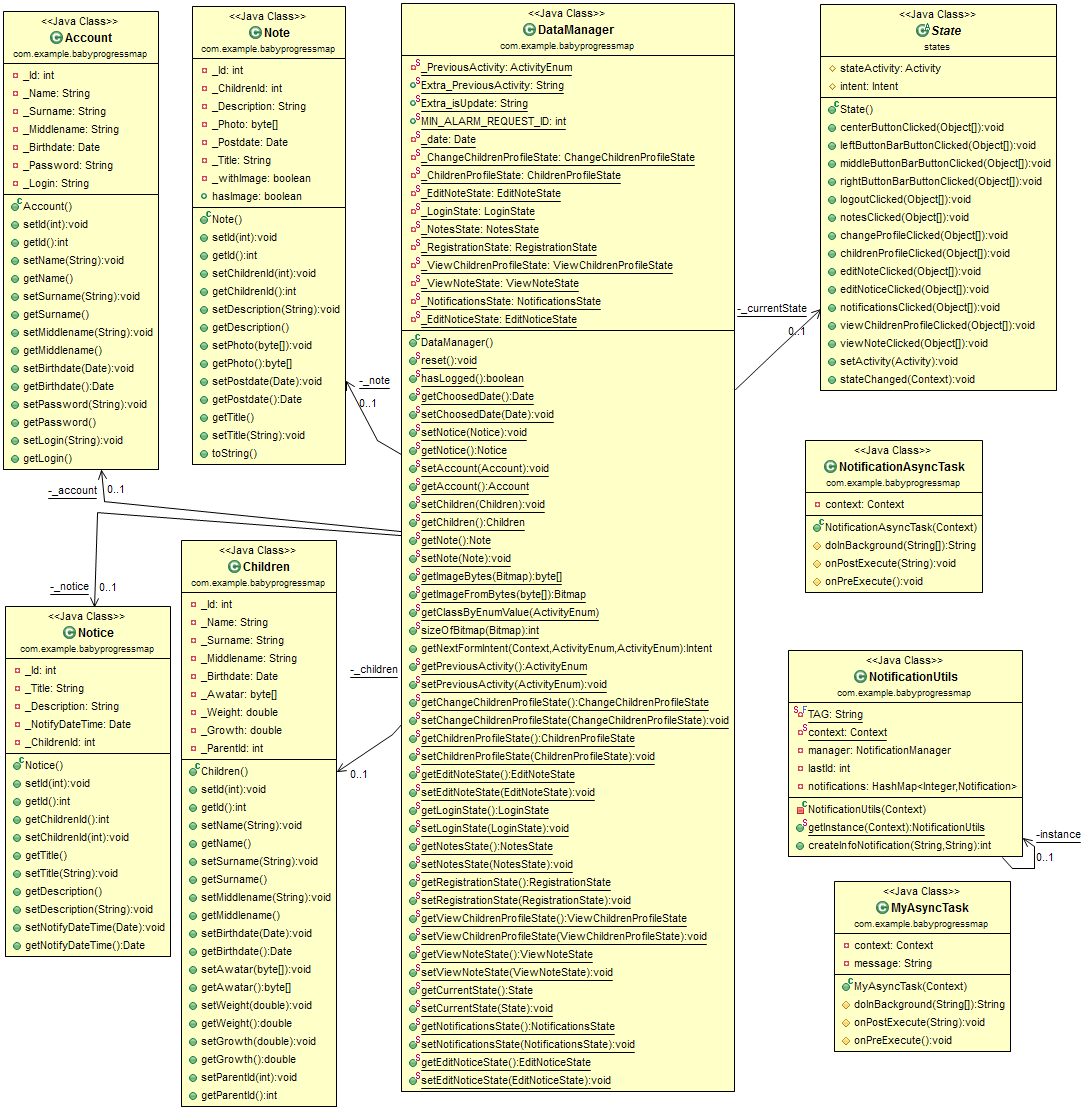


Рисунок - Диаграмма классов без взаимодействия с базой данных

## Разработка интерфейса

Графический интерфейс пользователя в Android основан на иерархии View и ViewGroup [18]. View объекты – это, как правило, графические виджеты, такие как кнопки, текстовые поля и невидимые контейнеры, которые определяют расположение или варианты использования дочерних элементов. Каждая из форм пользовательского интерфейса Android состоит из макета – файл определяющий набор элементов управления на форме – и класса, который содержит логику формы. Логика формы приложения состоит из следующих действий:

* загрузка элементов управления;
* реакция на взаимодействие пользователя с элементами управления;
* реакция на нажатие кнопок мобильного устройства;
* прием и передача данных между формами.

В процессе разработки графического интерфейса мобильного приложения были разработаны формы мобильного приложения.

**Форма аутентификации** (рисунок 18) – форма необходима для входа пользователем в учетную запись.

Возможные переходы:

* форма регистрации – в случае, если введенного пользователем имени учетной записи не существует;
* форма выбора профиля ребенка – в случае, если пользователь ввел корректные данные.

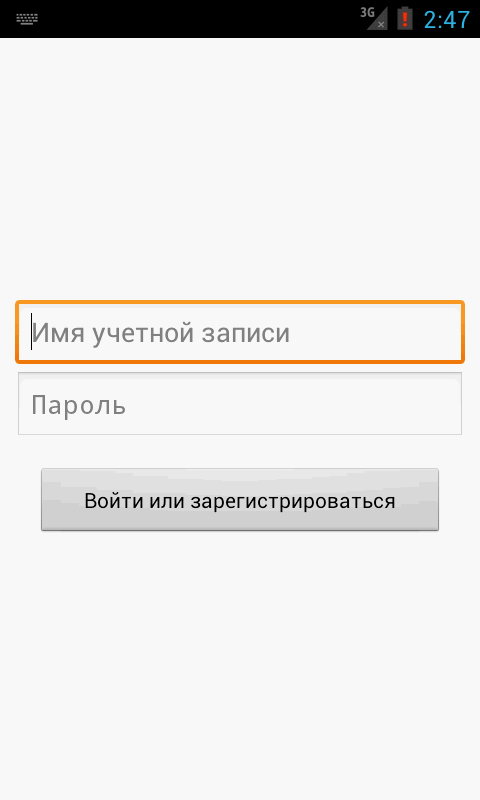


Рисунок - Форма аутентификации

**Форма регистрации** (рисунок 19) – форма необходима для регистрации пользователем новой учетной записи.

Возможные переходы:

* форма редактирования профиля ребенка – переход осуществляется в случае, если пользователем были введены корректные данные и новая учетная запись успешно создана;
* форма аутентификации – только после нажатия пользователем кнопки «назад».

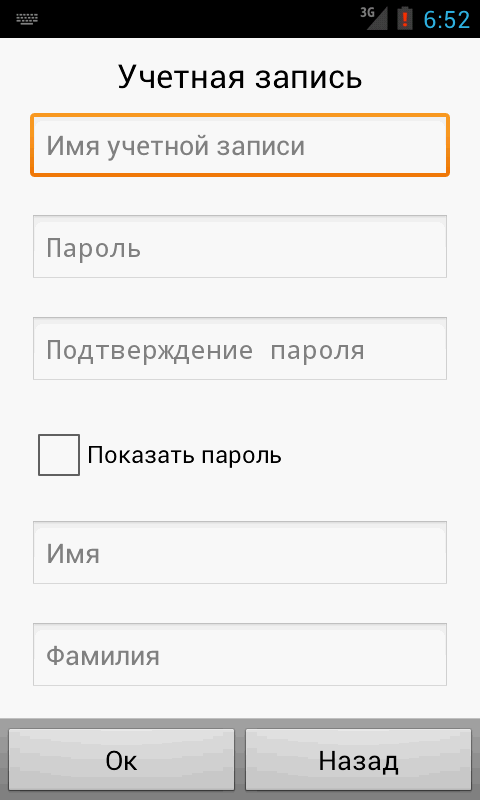


Рисунок - Форма регистрации

**Форма редактирования профиля ребенка** (рисунок 20) – форма необходима для внесения изменений в данные о ребенка.

Возможные переходы:

* форма выбора профиля ребенка – переход происходит в случае ввода пользователем корректных данных и подтверждения изменения профиля ребенка нажатием кнопки «ок»;
* форма выбора профиля ребенка – при условии, что эта форма была предыдущей и нажата кнопка «назад»;
* форма просмотра профиля ребенка – при условии, что эта форма была предыдущей и нажата кнопка «назад».

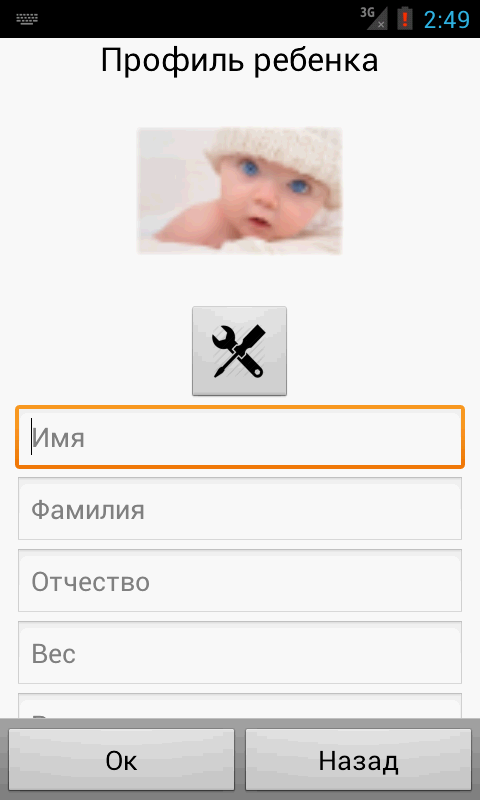


Рисунок – Форма редактирования профиля ребенка

**Форма просмотра профиля ребенка** (рисунок 21) – используется для просмотра данных, содержащихся в текущем профиле ребенка.

Возможные переходы:

* форма редактирования профиля ребенка – при нажатии на кнопку «изменить»;
* форма выбора профиля ребенка:
  + при нажатии на кнопку «Выбор профиля» элемента управления ActionBar;
  + при нажатии на кнопку «назад», при условии, что форма выбора профиля ребенка была предыдущей;
* форма просмотра заметок на дату:
  + при нажатии на кнопку «Заметки» элемента управления ActionBar;
  + при нажатии на кнопку «назад», при условии, что форма выбора заметок была предыдущей.



Рисунок – Форма просмотра профиля ребенка

**Форма профиля ребенка** (рисунок 22) – используется для выбора текущего профиля а также для создания нового профиля ребенка.

Возможные переходы:

* форма редактирования профиля ребенка – при нажатии на кнопку добавления нового профиля;
* форма просмотра заметок на дату – при нажатии на кнопку «заметки» элемента управления ActionBar;
* форма аутентификации – при нажатии на кнопку «выход» элемента управления ActionBar.

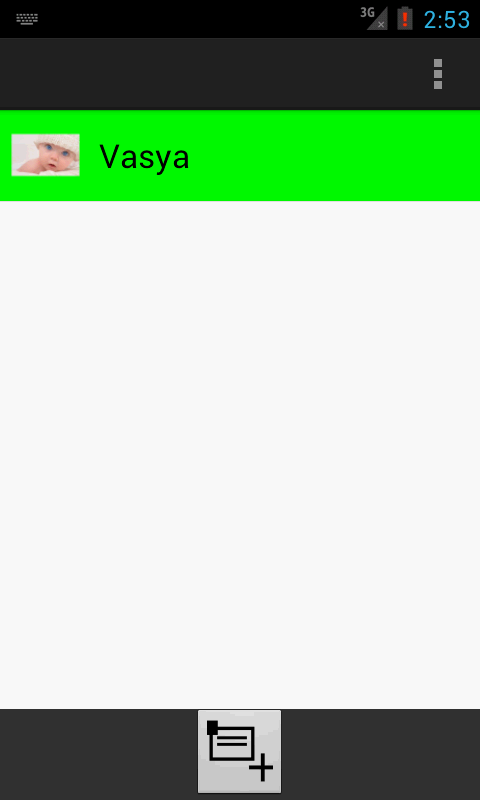


Рисунок – Форма выбора профиля ребенка

**Форма просмотра заметок на дату** (рисунок 23) – форма служит для просмотра добавленных на определенную дату заметок и для добавления новой заметки.

Возможные переходы:

* форма просмотра профиля ребенка – при нажатии на кнопку «профиль ребенка» элемента управления ActionBar;
* форма выбора профиля ребенка – при нажатии на кнопку «выбор профиля» элемента управления ActionBar;
* форма аутентификации - при нажатии на кнопку «выход» элемента управления ActionBar;
* форма редактирования заметки – при нажатии на кнопку добавления заметки.

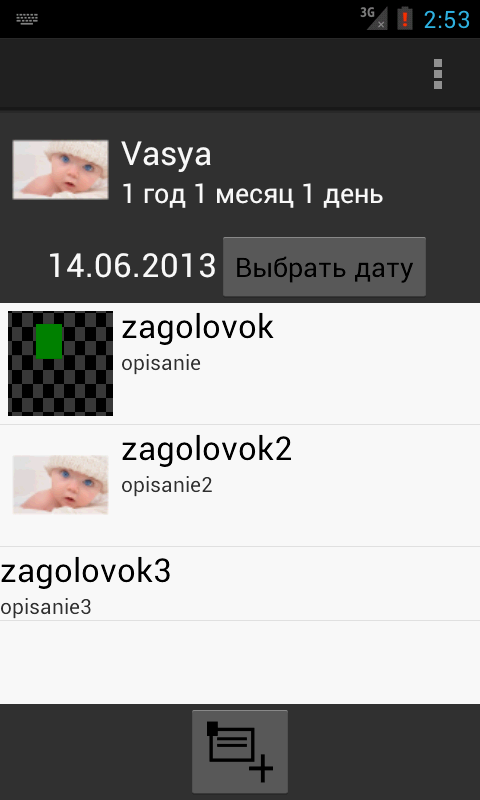


Рисунок – Форма просмотра заметок на дату

**Форма просмотра заметки** (рисунок 24) – форма служит для отображения заметки на весь экран.

Возможные переходы:

* форма просмотра заметок на дату – при нажатии на кнопку «назад»;
* форма редактирования заметки – при нажатии на кнопку «изменить»;
* форма просмотра заметок на дату – при нажатии на кнопку «заметки» элемента управления ActionBar;
* форма выбора профиля ребенка – при нажатии на кнопку «выбор профиля» элемента управления ActionBar;
* форма аутентификации – при нажатии на кнопку «выход» элемента управления ActionBar.

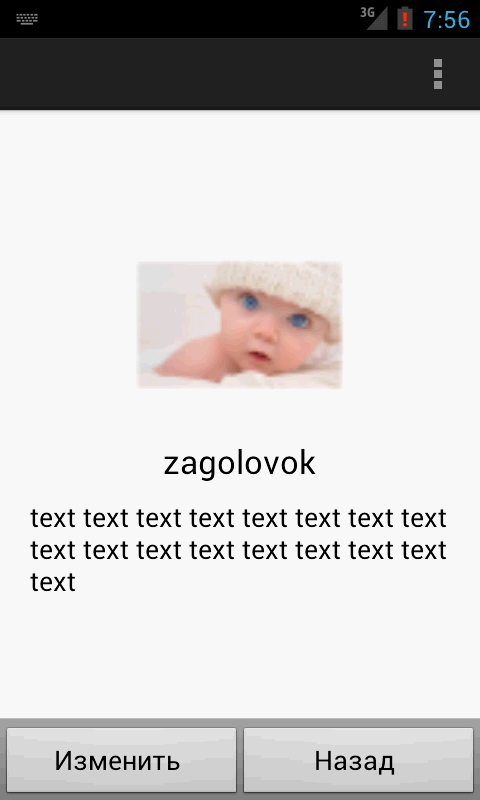


Рисунок - Форма просмотра заметки

**Форма редактирования заметки** (рисунок 25) – служит для внесения изменений в выбранную заметку или добавления новой.

Возможные переходы:

* форма просмотра заметок на дату – при нажатии на кнопку «назад» или «ок» и при условии, что эта форма была предыдущей;
* форма просмотра заметки – при нажатии на кнопку «назад» или «ок» и при условии, что эта форма была предыдущей.

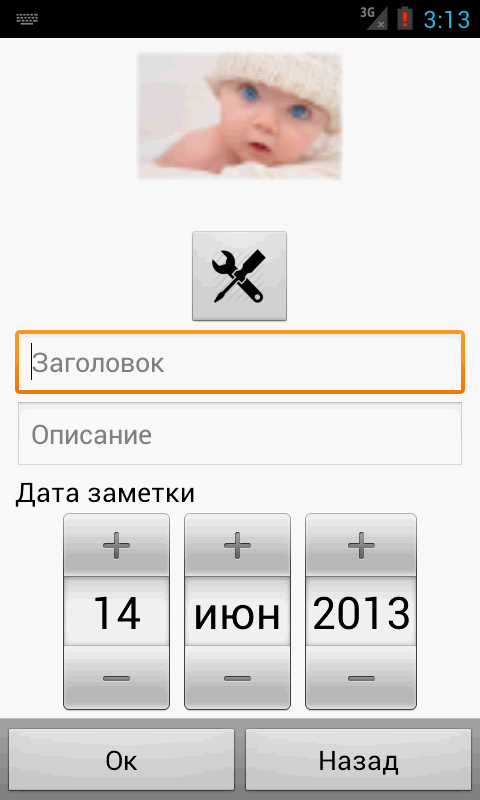


Рисунок - Форма редактирования заметок

**Форма редактирования уведомлений** (рисунок 26) – служит для добавления нового уведомления или внесения изменений в уже имеющиеся.

Возможные переходы:

* форма просмотра уведомлений – при нажатии на кнопку «ок» или кнопку «назад».

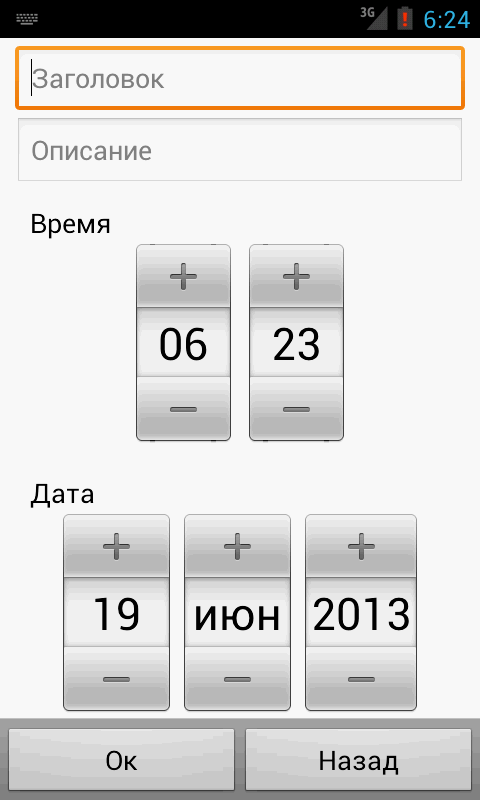


Рисунок - Форма редактирования уведомлений

**Форма просмотра уведомлений** (рисунок 27) – форма служит для просмотра и добавления уведомлений на определенные дату и время.

Возможные переходы:

* форма просмотра профиля ребенка – при нажатии на кнопку «Профиль ребенка» элемента управления ActionBar;
* форма выбора профиля ребенка – при нажатии на кнопку «Выбор профиля» элемента управления ActionBar;
* форма просмотра заметок на дату – при нажатии на кнопку «Заметки» элемента управления ActionBar;
* форма аутентификации – при нажатии на кнопку «Профиль ребенка» элемента управления ActionBar.

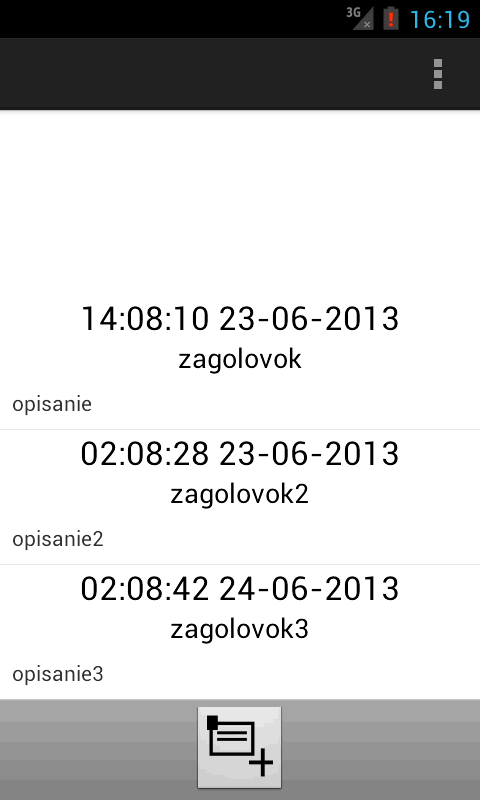


Рисунок – Форма просмотра уведомлений

# Тестирование программного обеспечения

## Тестирование

Тестирование программы – это процесс, целью которого является выявление дефектов и установление корректности работы анализируемой программы. Также в ходе тестирования необходимо выявить потенциальные источники появления ошибок в программном коде. Самый простой способ сделать это: перебрать все возможные варианты вводимых данных и проверить реакцию программы на эти данные. Этот метод не подходит для информационных систем большого масштаба, так как требует составление большого количества контрольных примеров и огромных трудозатрат. Но, в случае с разработанным в данной работе мобильным приложением, этот метод тестирования программного обеспечения полностью себя оправдывает. Ниже перечислены методики тестирования, которые актуальны при проверке работоспособности разработанного мобильного приложения:

1. Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функций, заложенных в программу.
2. Тестирование производительности – это тестирование программного обеспечения, нацеленное на проверку быстроты работы при выполнении заложенных в программу функций.
3. Тестирование пользовательского интерфейса – в контексте разработки мобильного приложения этот вид тестирования применяется для проверки корректного отображения пользовательского интерфейса на мобильных устройствах с экранами разных размеров.

## Функциональное тестирование

В основе разрабатываемого мобильного приложения лежат функции по взаимодействию с базой данных, функции по работе с изображениями и функции, обеспечивающие корректные переходы по формам приложения. Тестирование функции по взаимодействию с базой данных будет изображено на примере тестирования формы регистрации и формы редактирования профиля ребенка, при этом, для выявления корректности работы процедур по взаимодействию с базой данных будет использован SQLite Database Browser. Ход тестирования будет представлен в виде нескольких скриншотов на момент начала и окончания тестирования.

Тестирование формы регистрации

На примере формы регистрации проведен тест добавления данных в базу данных мобильного приложения. Для этого инициирована процедура регистрация с использованием тестовых данных.

Из рисунка (рисунок 28) видно, что на момент начала тестирования в таблице «account» отсутствовали какие-либо данные об учетных записях.

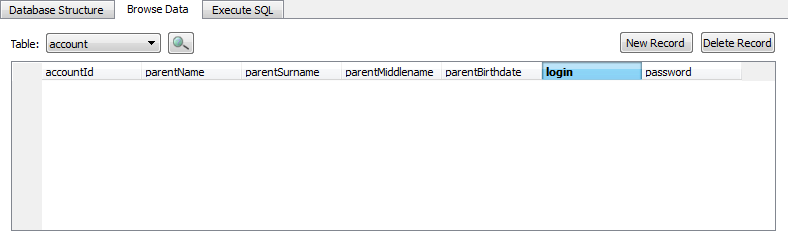


Рисунок – Данные таблицы «account»

Далее была использована форма регистрации для создания новой учетной записи (рисунок 29).

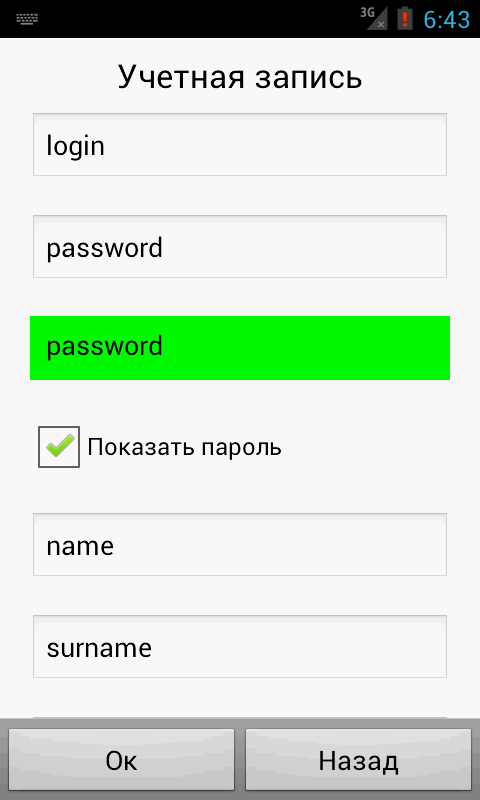


Рисунок - Регистрация новой учетной записи

На рисунке (рисунок 28) видно, что после создания новой учетной записи с использованием формы регистрации данные были успешно добавлены в таблицу базы данных.

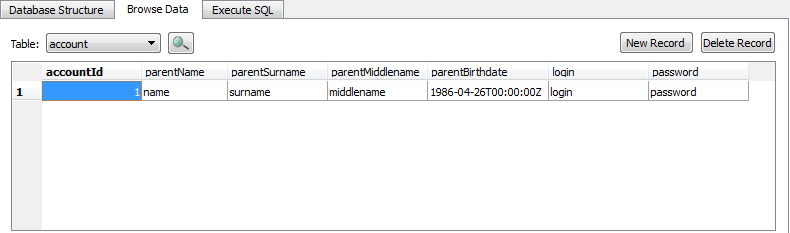


Рисунок - Данные таблицы "account"

Тестирование формы редактирования профиля ребенка

На примере формы редактирования профиля ребенка проведено тестирование процедур по изменению данных, содержащихся в базе данных, а также тестирование функции по работе с изображениями. На момент начала тестирования в таблице «children» уже имелась строка, содержащая информацию о профиле ребенка. Эти данные можно увидеть на следующем рисунке (рисунок 31):

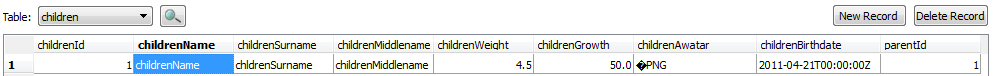


Рисунок - Данные таблицы "children"

Для проверки функций по изменению данных была использована форма редактирования профиля ребенка со следующими тестовыми данными (рисунок 32):

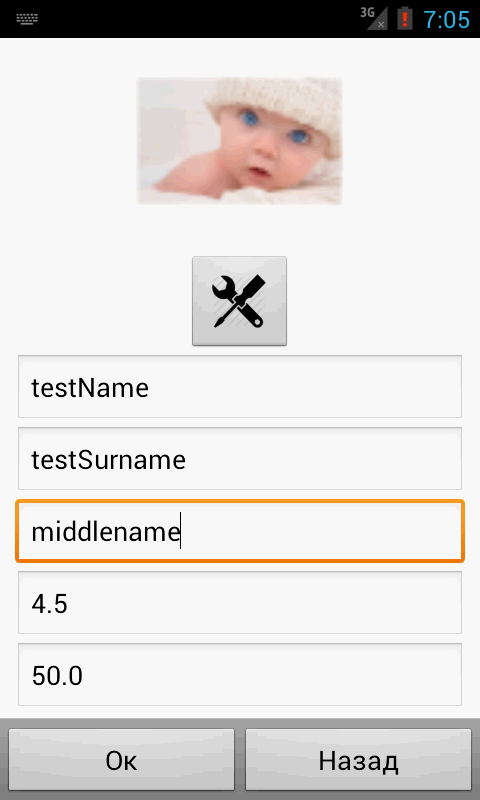


Рисунок - Редактирование существующего профиля ребенка

После применения изменений на рисунке (рисунок 33) можно увидеть, что данные таблицы «children» успешно изменены.

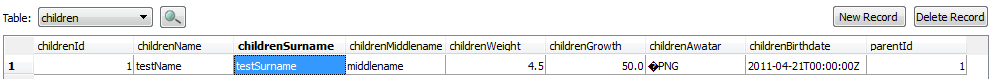


Рисунок - Данные таблицы "children"

В итоге проведения тестирования можно заключить о корректности работы функций по изменению данных таблицы. Также можно заключить о корректности выполнения процедур по работе с изображениями исходя из следующих фактов:

* изображение успешно помещено в таблицу базы данных в виде массива байт;
* массив байт успешно извлечен из базы данных и отображен на форме в виде полноценного изображения.

## Тестирование производительности

Для проведения тестирования производительности разработанного мобильного приложения использовалась программа DDMS. Это программа от компании Google, использующаяся для диагностики ресурсов, потребляемых мобильным приложением. В качестве основных функций программы можно выделить следующие:

* сервис перенаправления портов;
* взаимодействием с файловой системой эмулятора мобильного устройства;
* создание скриншотов экрана эмулятора;
* информирование о запущенных процессах и потоках эмулятора;
* логирование всех выполняющихся процессов;
* эмуляция входящих звонков и СМС;
* информирование о потребляемых запущенными процессами ресурсах.

Мониторинг потоков

Данный вид мониторинга предоставляет информацию о выполняющихся потоках конкретного процесса, в данном случае разработанного мобильного приложения. Выполняющиеся в приложении потоки изображены на следующем рисунке (рисунок 34):

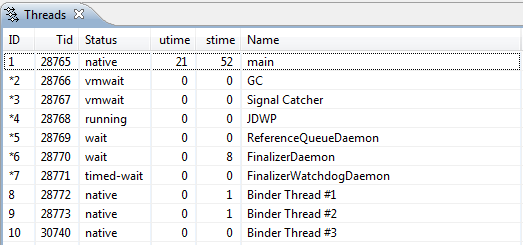


Рисунок - Окно мониторинга потоков

Окно мониторинга потоков процесса представляет собой таблицу, каждая запись которой отображает основную характеристику потока. Основными системными потоками каждого процесса являются следующие [33]:

* main – основной поток, протекающий в процессе;
* GC – поток, отвечающий за «сборку мусора»;
* Signal Catcher – поток, отвечающий за работу по приему сигнала;
* JDWP – поток, отвечающий за работу протокола взаимодействия среды разработки и виртуальной машины Java.

Описание полей окна мониторинга можно увидеть на следующей таблице (таблица 7):

Таблица - Описание полей окна мониторинга потоков

| ***Поле*** | ***Описание*** |
| --- | --- |
| ID | Уникальный идентификатор потока, выданный виртуальной машиной Java |
| Tid | Уникальный идентификатор потока, выданный ядром Linux |
| Status | Текущий статус потока |
| utime | Общее время, затраченное на выполнение пользовательского кода (единица измерения jiffy[[1]](#footnote-1)) |
| stime | Общее время, затраченное на выполнение системного кода (единица измерения jiffy) |
| Name | Наименование потока |

Статус потока позволяет определить разновидность работы потока в текущий момент времени. В каждый момент времени статус потока может представлять собой значение из следующего перечня [34]:

* running – выполняется код приложения;
* sleeping – поток в режиме сна вследствие вызова метода Thread.sleep();
* monitor – ожидание блокировки монитора;
* wait – поток в режиме ожидания вследствие вызова метода Object.wait();
* native – поток выполняет собственный код;
* vmwait – поток в режиме ожидания ресурсов виртуальной машины;
* zombie – поток находится в режиме завершения;
* init – поток находится в режиме инициализации;
* starting – поток находится в режиме запуска;

Из рисунка (рисунок 34) видно, что на момент проведения тестирования значительно преобладает выполнение системного кода. Суммарное время выполнения пользовательского кода составляет всего около 210 миллисекунд. Отсюда можно сделать вывод, что пользовательский код программы не требователен к ресурсам и обеспечивает высокую производительность.

Мониторинг распределения оперативной памяти

Данный вид мониторинга позволяет выявить значение используемой приложением оперативной памяти в общем и по каждому виду распределения в частности (рисунок 35).

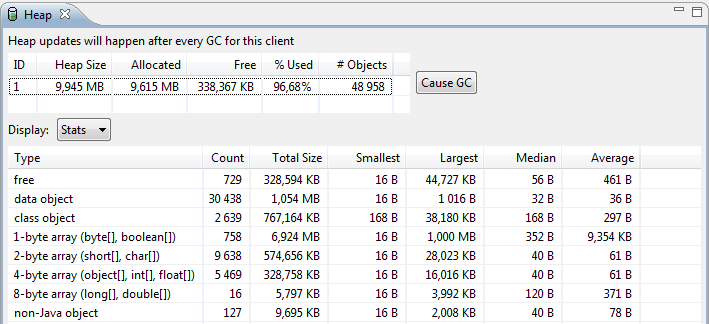


Рисунок - Окно мониторинга распределения оперативной памяти

Общий объем используемой оперативной памяти приложением отображается в верхней части окна, в поле «Heap Size» [31]. На данном рисунке (рисунок 34) видно, что общий объем оперативной памяти, потребляемый приложением, составляет около 10 мегабайт. Данное значение потребляемой оперативной памяти объясняется тем, значительная часть памяти требуется на обработку изображений, которые хранятся в виде массивов байт.

## Тестирование пользовательского интерфейса

Данный вид тестирования чрезвычайно важен при разработке мобильных приложений. Это объясняется тем, что мобильные устройства оснащаются экранами с разным соотношением сторон и плотностью пикселей. Поэтому графический интерфейс приложения должен быть гибким, чтобы корректно отображаться на экранах, обладающих разными размерами [21]. Тестирование пользовательского интерфейса проведено на примере формы редактирование профиля ребенка, так как эта форма имеет наибольше число элементов управления, по сравнению с другими формами разработанного приложения. Для проведения тестирования использовался функционал IDE Eclipse, позволяющий отобразить верстку формы на экране выбранного размера. В ходе тестирования проверке подвергалась корректность отображения элементов управления формы на экранах следующего размера:

* 10.1" (1280х800 mdpi) (рисунок 36)
* 4.0" (480х800 hdpi) (рисунок 37)
* 3.3" (240x480 ldpi) (рисунок 38)
* 2.7" (240x320 ldpi) (рисунок 39)

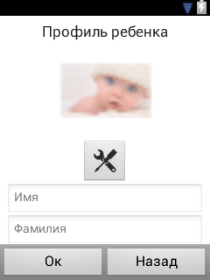


Рисунок - Тестируемая форма на экране 2.7"

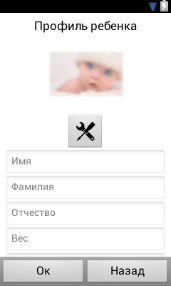


Рисунок - Тестируемая форма на экране 3.3"

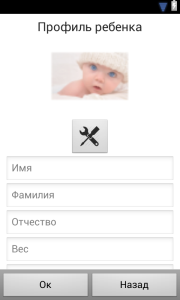


Рисунок - Тестируемая форма на экране 4.0"



Рисунок - Тестируемая форма на экране 10.1"

На основе изображений формы на различных экранах можно заметить, что на экранах с малыми и средними размерами не отображены сразу все элементы управления. Данный недостаток компенсируется тем, что среди элементов верстки формы использован элемент ScrollView. Этот элемент позволяет совершать скроллинг[[2]](#footnote-2) формы вверх и вниз. Вывод: данная форма может корректно отображаться на мобильных устройствах с различными размерами экрана. Данный вывод применим и к остальным формам разработанного приложения, так как при их верстке также учитывался принцип масштабируемости интерфейса.

## Перспективы

В перспективе развития разработанного мобильного приложения планируется реализовать функцию синхронизации данных с сервером. Под синхронизацией данных понимается создание резервной копии базы данных приложения на удаленном сервере. Помимо создания резервной копии данных, эта функция должна предоставить возможность копирования данных с удаленного сервера в локальную базу данных приложения. Также к перспективе развития программы можно отнести создание более привлекательного интерфейса и создания дополнительных опций просмотра данных. В ходе дальнейшего развития разработанного приложения планируется публикация программы в интернет - магазине мобильных приложений GooglePlay.

# ЗАКЛЮЧЕНИЕ

Сфера мобильных технологий глубоко проникла в нашу жизнь. И уже сложно представить прожитый день без одного из технологических устройств. При этом количество проданных устройств с каждым днем только увеличивается. И многие из них находят себе применение в самых различных сферах нашей жизни. Однако стоит заметить, что без большого выбора приложений мобильные устройства не стали бы так популярны. Именно поэтому, настолько актуальной сегодня является сфера разработки мобильных приложений.

В данной работе в качестве объекта исследования было принято мобильное приложение для операционной системы Android. Предметом исследования является приложение для управления данными о развитии ребенка для мобильного устройства. Цель работы - разработка нативного мобильного приложения, ориентированного на мобильную операционную систему Android. В качестве основных инструментов реализации мобильного приложения были выбраны средства для проектирования (Microsoft Visio, DeZign for Databases) и непосредственной разработки (Eclipse, ADT, SQLite Database Manager) продукта проекта. Также был проведен анализ предметной области – рынка мобильных приложений.

В процессе проектирования базы данных приложения были выявлены объекты, необходимые в последующем для реализации базы данных. Для каждого из объектов определен набор атрибутов, однозначно его характеризующих. На основании полученных данных были созданы логическая и физическая модели базы данных. Реализация базы данных совершена на основе моделей и с использованием функционала языка программирования Java.

Для написания программного кода программы составлена архитектура необходимых классов, составлены диаграммы классов, которые в последствии были реализованы на языке программирования Java.

При проектировании форм приложения автор руководствовался принципами простоты и эргономичности интерфейса – на формах приложения содержится наименьше количество элементов управления. Это нужно для того, чтобы интерфейс не казался пользователю перегруженным, а управление приложением сложным. Для разработки форм и макетов приложения был использован язык разметки XML, а для удобной навигации по формам приложения созданы меню элемента управления ActionBar. В целях избежания ошибок и неправльной работы форм, во всех формах приложения отключена реакция на нажатие аппаратной кнопки «Назад».

В ходе тестирования приложения были проведены тесты функциональности, производительности и пользовательского интерфейса. При проведении тестов функциональности замечено, что никаких сбоев в работе приложения нет, а все переходы по формам совершаются корректно. На основании проведенных тестов производительности было выявлено, что приложение нетребовательно к аппаратным ресурсам мобильного устройства. Также, на основании тестирования пользовательского интерфейса было выявлено, что формы разработанного приложения корректно отображаются на мобильных устройствах с разными размерами экрана и соотношениями сторон экрана.

В результате выполнения данной выпускной квалификационной работы, на основании требований к программе и работ по проектированию программного обеспечения, было создано приложение для мобильного устройства. Созданное приложение позволяет вести историю развития ребенка или нескольких детей. Основной функцией приложения, которая и позволяет вести историю развития ребенка, является функция добавления заметок в базу данных приложения. Эти заметки позволят сохранить памятные моменты из жизни ребенка на мобильном устройстве. Также заметки могут быть использованы для сохранения фотографий и описания к ним. Помимо заметок, в результате разработки программы реализована возможность добавления уведомлений на определенную дату. При наступлении даты и времени, которые указал пользователь, происходит стандартная процедура оповещения. При стандартной процедуре оповещения происходит воспроизведение звукового сигнала, включается диодное оповещение и появляется сообщение (заготовленное пользователем) в панели состояний мобильного устройства. Данная функция позволит пользователю приложения не забывать о каких-либо важных для него событиях.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Android лидирует среди мобильных ОС в Европе. [Электронный документ] URL: <http://news.eizvestia.com/news_technology/full/android-lidiruet-sredi-mobilnyh-os-v-evrope>
2. App Store – Официальный сайт Apple [Электронный документ] URL: http://store.apple.com/ru.
3. Eclipse (среда разработки). [Электронный документ] URL: <http://ru.wikipedia.org/wiki/Eclipse_(%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)>
4. Google Play. [Электронный документ] URL: <https://play.google.com/intl/ru/about/>
5. IBM – Один из крупнейших в мире производителей и поставщиков аппаратного и программног обеспечения, а также ИТ – сервисов и консальтинговых услуг. [Электронный документ] URL: <http://ru.wikipedia.org/wiki/IBM>.
6. IDC – аналитическая фирма, специализирующаяся на исследованиях рынка [информационных технологий](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8). [Электронный документ] URL: <http://ru.wikipedia.org/wiki/International_Data_Corporation>.
7. J’son & Partners Consulting – ведущее консалтинговое агентство на рынках телекоммуникаций, ИТ, Медиа и инновационных технологий. [Электронный документ] URL: <http://www.json.ru/>.
8. Барри Берд. Программирование на Java для чайников. Диалектика, 2013. 384 с.
9. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирование. Паттерны проектирования. СПб.: Питер, 2007. 366 с.
10. Герберт Шилдт. Java. Полное руководство. Вильямся, 2012, 1104 с.
11. Джейсон Мак-Колм Смит. Элементарные шаблоны проектирования. Вильямс, 2013. 304 с.
12. Джен Л. Харрингтон. Проектирование реляционных баз данных. Лори, 2006. 210 с.
13. Джон Влиссидес – Применение шаблонов проектирования. Дополнительные штрихи. Вильямс, 2003. 144 с.
14. Донн Фелкер. Android: разработка приложений для чайников. Диалектика, Вильямс, 2012. 336 с.
15. Кренке Д. Теория и практика построения баз данных. СПб.: Питер, 2005. 864 с.
16. Магазин приложений + игр Windows Phone. [Электронный ресурс] URL: <http://www.windowsphone.com/ru-ru/store>
17. Майкл Морган. Java 2. Руководство разработчика. Вильямс, 2000. 720 с.
18. Рид Роджерс, Джон Ломбардо, Зигурд Медниекс, Блейк Мейк – Android. Разработка приложений. ЭКОМ Паблишерз, 2010. 400 с.
19. Робер Седжвик, Кевин Уэйн. Алгоритмы на Java. Вильямс, 2012. 848 с.
20. Рынок мобильных приложений к 2016 году вырастет до 65,79 млрд долларов - Юлия Воронина - Российская газета [Электронный документ] URL: <http://www.rg.ru/2013/03/19/prilizenia.html>.
21. Сатия Коматинени, Дэйв Маклин, Саид Хашими – Android 3 для профессионалов. Создание приложений для планшетных компьютеров и смартфонов. Вильямс, 2012. 1024 с.
22. Стивен Макконнел. Совершенный код. – Санкт-Петербург: Питер, 2007. 896 c.
23. Стивен Стелтинг, Олав Маасен. Применение шаблонов Java. Библиотека профессионала. Вильямс, 2002. 576 с.
24. Стивен Хольцнер. Eclipse. Вильямс, 2009. 334 с.
25. Тимоти Бадд. Объектно-ориентированное программирование в действии. СПб.: Питер, 1997. 366 с.
26. Фримен Эр., Фримен Эл., Сьерра К., Бейтс Б. Паттерны проектирования. СПб.: Питер, 2011. 645 c.
27. Хашими С., Коматинени С., Маклин Д. Разработка приложений для Android. СПб.: Питер, 2011. 735 с.
28. Эрик Эванс. Предметно-ориентированное проектирование. Вильямс, 2010. 448 с.
29. Android Developers [Электронный ресурс] URL: [http://developer.android.com](http://developer.android.com/).
30. BlackBerry World – Free & Paid BlackBerry Apps for Smartphones & Tablets. [Электронный ресурс] URL: <http://appworld.blackberry.com/webstore/?>
31. Deepak Vohra – Java EE Development with Eclipse. Packt Publishing, 2012. 426 с.
32. More Than 50% Using Mobile Phones During Shopping. [Электронный документ] URL: <http://internet2go.net/news/data-and-forecasts/global-survey-offers-more-data-shopping-and-smartphones>
33. Oracle Documentation [Электронный ресурс] URL: <http://docs.oracle.com/>.
34. Stack Overflow [Электронный ресурс] URL: <http://stackoverflow.com/>.
35. Оплатить проезд в общественном транспорте с помощью мобильного телефона. [Электронный документ] URL: <http://www.bilet77msk.ru/news-center-uslug/news-bileti-na-metro-aeroekspress-oplatotj-po-mobiljnomu-telefonu.html>

# ПРИЛОЖЕНИЯ

#### ПРИЛОЖЕНИЕ А

Код классов из пакета «com.example.babyprogressmap»

Код класса «Account»

package com.example.babyprogressmap;

import java.util.Date;

public class Account {

public Account() {

\_Id = -1;

\_Name = "";

\_Surname = "";

\_Middlename = "";

\_Birthdate = new Date();

\_Password = "";

\_Login = "";

}

private int \_Id;

private String \_Name;

private String \_Surname;

private String \_Middlename;

private Date \_Birthdate;

private String \_Password;

private String \_Login;

public void setId(int newId) {

\_Id = newId;

}

public int getId() {

return \_Id;

}

public void setName(String newName) {

\_Name = newName;

}

public String getName() {

return \_Name;

}

public void setSurname(String newSurname) {

\_Surname = newSurname;

}

public String getSurname() {

return \_Surname;

}

public void setMiddlename(String newMiddlename) {

\_Middlename = newMiddlename;

}

public String getMiddlename() {

return \_Middlename;

}

public void setBirthdate(Date newBirthdate) {

\_Birthdate = newBirthdate;

}

public Date getBirthdate() {

return \_Birthdate;

}

public void setPassword(String newPassword)

{

\_Password = newPassword;

}

public String getPassword()

{

return \_Password;

}

public void setLogin(String newLogin)

{

\_Login = newLogin;

}

public String getLogin()

{

return \_Login;

}

}

1. Код класса «ActivityEnum»

package com.example.babyprogressmap;

public enum ActivityEnum {

Login, Registration, ChildrenProfile, ChangeChildrenProfile, Notes, ViewChildrenProfile, ViewNote, EditNote, EditNotice, Notifications, Null

}

1. Код класса «AlarmReceiver»

package com.example.babyprogressmap;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.os.Bundle;

import android.widget.Toast;

public class AlarmReceiver extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent) {

try {

Bundle bundle = intent.getExtras();

String message = bundle.getString("alarm\_message");

String title = bundle.getString("title");

new NotificationAsyncTask(context).doInBackground(title, message);

} catch (Exception e) {

}

}

}

1. Код класса «BabyProgressDataBaseHelper»

package com.example.babyprogressmap;

import java.io.FileNotFoundException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.util.Log;

public class BabyProgressDataBaseHelper extends SQLiteOpenHelper {

private static BabyProgressDataBaseHelper instance = null;

public static final String DATABASE\_NAME = "BabyProgressDataBase";

private static final int DATABASE\_VERSION = 1;

public static final String CHILDREN\_TABLE\_NAME = "children";

public static final String CHILDREN\_ID = "childrenId";

public static final String CHILDREN\_NAME = "childrenName";

public static final String CHILDREN\_SURNAME = "childrenSurname";

public static final String CHILDREN\_MIDDLENAME = "childrenMiddlename";

public static final String CHILDREN\_WEIGHT = "childrenWeight";

public static final String CHILDREN\_GROWTH = "childrenGrowth";

public static final String CHILDREN\_AWATAR = "childrenAwatar";

public static final String CHILDREN\_BIRTHDATE = "childrenBirthdate";

public static final String CHILDREN\_PARENT\_ID = "parentId";

public static final String ACCOUNT\_TABLE\_NAME = "account";

public static final String ACCOUNT\_ID = "accountId";

public static final String ACCOUNT\_PARENT\_NAME = "parentName";

public static final String ACCOUNT\_PARENT\_SURNAME = "parentSurname";

public static final String ACCOUNT\_PARENT\_MIDDLENAME = "parentMiddlename";

public static final String ACCOUNT\_PARENT\_BIRTHDATE = "parentBirthdate";

public static final String ACCOUNT\_LOGIN = "login";

public static final String ACCOUNT\_PASSWORD = "password";

public static final String NOTE\_TABLE\_NAME = "note";

public static final String NOTE\_ID = "noteID";

public static final String NOTE\_DESCRIPTION = "noteDescription";

public static final String NOTE\_POSTDATE = "notePostdate";

public static final String NOTE\_CHILDREN\_ID = "childrenId";

public static final String NOTE\_PHOTO = "notePhoto";

public static final String NOTE\_TITLE = "noteTitle";

public static final String NOTE\_WITH\_IMAGE = "noteWithImage";

public static final String NOTICE\_TABLE\_NAME = "notice";

public static final String NOTICE\_ID = "noticeId";

public static final String NOTICE\_NOTIFYDATETIME = "notifyDateTime";

public static final String NOTICE\_TITLE = "noteTitle";

public static final String NOTICE\_DESCRIPTION = "noteDescription";

public static final String NOTICE\_CHILDREN\_ID = "noteChildrenId";

public static final String CREATE\_TABLE\_NOTICE = "create table "

+ NOTICE\_TABLE\_NAME + "(" + NOTICE\_ID

+ " integer primary key autoincrement," + NOTICE\_NOTIFYDATETIME

+ " text," + NOTICE\_TITLE + " text," + NOTICE\_DESCRIPTION

+ " text," + NOTICE\_CHILDREN\_ID + " integer," + "foreign key("

+ NOTICE\_CHILDREN\_ID + ") references " + CHILDREN\_TABLE\_NAME + "("

+ CHILDREN\_ID + "));";

public static final String CREATE\_TABLE\_CHILDREN = "create table "

+ CHILDREN\_TABLE\_NAME + "(" + CHILDREN\_ID

+ " integer primary key autoincrement," + CHILDREN\_NAME + " text,"

+ CHILDREN\_SURNAME + " text," + CHILDREN\_MIDDLENAME + " text,"

+ CHILDREN\_WEIGHT + " real," + CHILDREN\_GROWTH + " real,"

+ CHILDREN\_AWATAR + " blob," + CHILDREN\_BIRTHDATE + " text,"

+ CHILDREN\_PARENT\_ID + " integer," + "foreign key("

+ CHILDREN\_PARENT\_ID + ") references " + ACCOUNT\_TABLE\_NAME + "("

+ ACCOUNT\_ID + "));";

public static final String CREATE\_TABLE\_ACCOUNT = "create table "

+ ACCOUNT\_TABLE\_NAME + "(" + ACCOUNT\_ID

+ " integer primary key autoincrement," + ACCOUNT\_PARENT\_NAME

+ " text," + ACCOUNT\_PARENT\_SURNAME + " text,"

+ ACCOUNT\_PARENT\_MIDDLENAME + " text," + ACCOUNT\_PARENT\_BIRTHDATE

+ " text," + ACCOUNT\_LOGIN + " text," + ACCOUNT\_PASSWORD

+ " text);";

public static final String CREATE\_TABLE\_NOTE = "create table "

+ NOTE\_TABLE\_NAME + "(" + NOTE\_ID

+ " integer primary key autoincrement," + NOTE\_DESCRIPTION

+ " text," + NOTE\_TITLE + " text," + NOTE\_POSTDATE + " text,"

+ NOTE\_PHOTO + " blob," + NOTE\_WITH\_IMAGE + " integer,"

+ NOTE\_CHILDREN\_ID + " integer,"

+ "foreign key(" + NOTE\_CHILDREN\_ID + ") references "

+ CHILDREN\_TABLE\_NAME + "(" + CHILDREN\_ID + "));";

private BabyProgressDataBaseHelper(Context context) {

super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

}

public static BabyProgressDataBaseHelper getInstance(Context context) {

if (instance == null) {

instance = new BabyProgressDataBaseHelper(context);

return instance;

} else

return instance;

}

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL("DROP TABLE IF EXISTS " + NOTICE\_TABLE\_NAME);

db.execSQL("DROP TABLE IF EXISTS " + NOTE\_TABLE\_NAME);

db.execSQL("DROP TABLE IF EXISTS " + CHILDREN\_TABLE\_NAME);

db.execSQL("DROP TABLE IF EXISTS " + ACCOUNT\_TABLE\_NAME);

onCreate(db);

}

public void onCreate(SQLiteDatabase database) {

database.execSQL(CREATE\_TABLE\_ACCOUNT);

database.execSQL(CREATE\_TABLE\_CHILDREN);

database.execSQL(CREATE\_TABLE\_NOTE);

database.execSQL(CREATE\_TABLE\_NOTICE);

}

}

1. Код класса «Children»

package com.example.babyprogressmap;

import java.text.SimpleDateFormat;

import java.util.Date;

public class Children {

private int \_Id;

private String \_Name;

private String \_Surname;

private String \_Middlename;

private Date \_Birthdate;

private byte[] \_Awatar;

private double \_Weight;

private double \_Growth;

private int \_ParentId;

public Children(){

\_Id = 0;

\_Name = "";

\_Surname = "";

\_Middlename = "";

\_Birthdate = new Date();

\_Awatar = null;

\_Weight = 0;

\_Growth = 0;

\_ParentId = -1;

}

public void setId(int newId) {

\_Id = newId;

}

public int getId() {

return \_Id;

}

public void setName(String newName) {

\_Name = newName;

}

public String getName() {

return \_Name;

}

public void setSurname(String newSurname) {

\_Surname = newSurname;

}

public String getSurname() {

return \_Surname;

}

public void setMiddlename(String newMiddlename) {

\_Middlename = newMiddlename;

}

public String getMiddlename() {

return \_Middlename;

}

public void setBirthdate(Date newBirthdate) {

\_Birthdate = newBirthdate;

}

public Date getBirthdate() {

return \_Birthdate;

}

public void setAwatar(byte[] newAwatar) {

\_Awatar = newAwatar;

}

public byte[] getAwatar() {

return \_Awatar;

}

public void setWeight(double newWeight) {

\_Weight = newWeight;

}

public double getWeight() {

return \_Weight;

}

public void setGrowth(double newGrowth) {

\_Growth = newGrowth;

}

public double getGrowth() {

return \_Growth;

}

public void setParentId(int newParentId) {

\_ParentId = newParentId;

}

public int getParentId() {

return \_ParentId;

}

}

1. Код класса «DataAdapter»

package com.example.babyprogressmap;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import android.annotation.SuppressLint;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

@SuppressLint("SimpleDateFormat")

public class DataAdapter {

private Context context;

private BabyProgressDataBaseHelper helper;

private SQLiteDatabase db;

private boolean \_isClosed = true;

public static final String DATE\_FORMAT\_RUS = "dd.MM.yyyy";

public static final String DATE\_FORMAT = "yyyy-MM-dd";

public static final String DATE\_TIME\_FORMAT = "yyyy-MM-dd'T'HH:mm:ss'Z'";

public static final String DATE\_TIME\_FORMAT\_REVERSED = "HH:mm:ss dd-MM-yyyy";

public DataAdapter(Context context) {

this.context = context;

}

public void open() {

helper = BabyProgressDataBaseHelper.getInstance(context);

db = helper.getWritableDatabase();

\_isClosed = false;

}

public void close() {

helper.close();

\_isClosed = true;

}

public boolean isClosed() {

return \_isClosed;

}

public long insertNote(Note note) {

ContentValues cv = new ContentValues();

cv.put(BabyProgressDataBaseHelper.NOTE\_CHILDREN\_ID,

note.getChildrenId());

Date dt = note.getPostdate();

SimpleDateFormat format = new SimpleDateFormat(DATE\_TIME\_FORMAT);

String resultDate = format.format(dt);

cv.put(BabyProgressDataBaseHelper.NOTE\_POSTDATE, resultDate);

cv.put(BabyProgressDataBaseHelper.NOTE\_DESCRIPTION,

note.getDescription());

if (note.hasImage)

cv.put(BabyProgressDataBaseHelper.NOTE\_PHOTO, note.getPhoto());

else

cv.put(BabyProgressDataBaseHelper.NOTE\_PHOTO, new byte[0]);

cv.put(BabyProgressDataBaseHelper.NOTE\_TITLE, note.getTitle());

int i = 0;

if (note.hasImage)

i = 1;

cv.put(BabyProgressDataBaseHelper.NOTE\_WITH\_IMAGE, i);

return db.insert(BabyProgressDataBaseHelper.NOTE\_TABLE\_NAME, null, cv);

}

public long insertParent(Account parent) {

ContentValues cv = new ContentValues();

Date bd = parent.getBirthdate();

SimpleDateFormat format = new SimpleDateFormat(DATE\_TIME\_FORMAT);

String result = format.format(bd);

cv.put(BabyProgressDataBaseHelper.ACCOUNT\_PARENT\_BIRTHDATE, result);

cv.put(BabyProgressDataBaseHelper.ACCOUNT\_PARENT\_MIDDLENAME,

parent.getMiddlename());

cv.put(BabyProgressDataBaseHelper.ACCOUNT\_PARENT\_NAME, parent.getName());

cv.put(BabyProgressDataBaseHelper.ACCOUNT\_PARENT\_SURNAME,

parent.getSurname());

cv.put(BabyProgressDataBaseHelper.ACCOUNT\_LOGIN, parent.getLogin());

cv.put(BabyProgressDataBaseHelper.ACCOUNT\_PASSWORD,

parent.getPassword());

return db.insert(BabyProgressDataBaseHelper.ACCOUNT\_TABLE\_NAME, null,

cv);

}

public long insertChildren(Children children) {

ContentValues cv = new ContentValues();

cv.put(BabyProgressDataBaseHelper.CHILDREN\_AWATAR, children.getAwatar());

Date dt = children.getBirthdate();

SimpleDateFormat format = new SimpleDateFormat(DATE\_TIME\_FORMAT);

String resultDate = format.format(dt);

cv.put(BabyProgressDataBaseHelper.CHILDREN\_BIRTHDATE, resultDate);

cv.put(BabyProgressDataBaseHelper.CHILDREN\_GROWTH, children.getGrowth());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_MIDDLENAME,

children.getMiddlename());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_NAME, children.getName());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_PARENT\_ID,

children.getParentId());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_SURNAME,

children.getSurname());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_WEIGHT, children.getWeight());

return db.insert(BabyProgressDataBaseHelper.CHILDREN\_TABLE\_NAME, null,

cv);

}

public long insertNotice(Notice notice) {

ContentValues cv = new ContentValues();

cv.put(BabyProgressDataBaseHelper.NOTICE\_CHILDREN\_ID,

notice.getChildrenId());

SimpleDateFormat format = new SimpleDateFormat(

DataAdapter.DATE\_TIME\_FORMAT);

String dateRes = format.format(notice.getNotifyDateTime());

cv.put(BabyProgressDataBaseHelper.NOTICE\_NOTIFYDATETIME, dateRes);

cv.put(BabyProgressDataBaseHelper.NOTICE\_TITLE, notice.getTitle());

cv.put(BabyProgressDataBaseHelper.NOTICE\_DESCRIPTION,

notice.getDescription());

return db

.insert(BabyProgressDataBaseHelper.NOTICE\_TABLE\_NAME, null, cv);

}

public Cursor getCursor(String tableName) {

String sql = "select \* from " + tableName;

return db.rawQuery(sql, new String[] {});

}

public ArrayList<Children> getChildrens() {

Cursor cur = getCursor(BabyProgressDataBaseHelper.CHILDREN\_TABLE\_NAME);

ArrayList<Children> list = new ArrayList<Children>();

while (cur.moveToNext()) {

Children child = new Children();

child.setId(cur.getInt(0));

child.setName(cur.getString(1));

child.setSurname(cur.getString(2));

child.setMiddlename(cur.getString(3));

child.setWeight(cur.getDouble(4));

child.setGrowth(cur.getDouble(5));

child.setAwatar(cur.getBlob(6));

SimpleDateFormat format = new SimpleDateFormat(DATE\_TIME\_FORMAT);

try {

Date dt = format.parse(cur.getString(7));

child.setBirthdate(dt);

} catch (ParseException e) {

e.printStackTrace();

}

child.setParentId(cur.getInt(8));

list.add(child);

}

return list;

}

public Children getChildrenById(int id) {

String sql = "select \* from "

+ BabyProgressDataBaseHelper.CHILDREN\_TABLE\_NAME + " where "

+ BabyProgressDataBaseHelper.CHILDREN\_ID + " = " + id;

Cursor cur = db.rawQuery(sql, new String[] {});

Children child = new Children();

if (cur.moveToFirst()) {

child.setId(cur.getInt(0));

child.setName(cur.getString(1));

child.setSurname(cur.getString(2));

child.setMiddlename(cur.getString(3));

child.setWeight(cur.getDouble(4));

child.setGrowth(cur.getDouble(5));

child.setAwatar(cur.getBlob(6));

SimpleDateFormat format = new SimpleDateFormat(DATE\_FORMAT);

try {

Date dt = format.parse(cur.getString(7));

child.setBirthdate(dt);

} catch (ParseException e) {

e.printStackTrace();

}

child.setParentId(cur.getInt(8));

}

return child;

}

public int getAccountId(String login) {

String sql = "select \* from "

+ BabyProgressDataBaseHelper.ACCOUNT\_TABLE\_NAME + " where "

+ BabyProgressDataBaseHelper.ACCOUNT\_LOGIN + " = '" + login+ "'";

Cursor cur = db.rawQuery(sql, new String[] {});

int id = -1;

if (cur.moveToFirst()) {

id = cur.getInt(0);

}

return id;

}

public ArrayList<Account> getAccounts() {

Cursor cur = getCursor(BabyProgressDataBaseHelper.ACCOUNT\_TABLE\_NAME);

ArrayList<Account> list = new ArrayList<Account>();

while (cur.moveToNext()) {

Account parent = new Account();

parent.setId(cur.getInt(0));

parent.setName(cur.getString(1));

parent.setSurname(cur.getString(2));

parent.setMiddlename(cur.getString(3));

SimpleDateFormat format = new SimpleDateFormat(DATE\_FORMAT);

try {

parent.setBirthdate(format.parse(cur.getString(4)));

} catch (ParseException e) {

e.printStackTrace();

}

parent.setLogin(cur.getString(5));

parent.setPassword(cur.getString(6));

list.add(parent);

}

return list;

}

public ArrayList<Notice> getNotices() {

Cursor cur = getCursor(BabyProgressDataBaseHelper.NOTICE\_TABLE\_NAME);

ArrayList<Notice> list = new ArrayList<Notice>();

while (cur.moveToNext()) {

Notice notice = new Notice();

notice.setId(cur.getInt(0));

SimpleDateFormat format = new SimpleDateFormat(DATE\_TIME\_FORMAT);

try {

notice.setNotifyDateTime(format.parse(cur.getString(1)));

} catch (ParseException e) {

e.printStackTrace();

}

notice.setTitle(cur.getString(2));

notice.setDescription(cur.getString(3));

list.add(notice);

}

return list;

}

public ArrayList<Note> getNotes() {

Cursor cur = getCursor(BabyProgressDataBaseHelper.NOTE\_TABLE\_NAME);

ArrayList<Note> list = new ArrayList<Note>();

while (cur.moveToNext()) {

Note note = new Note();

note.setId(cur.getInt(0));

note.setDescription(cur.getString(1));

note.setTitle(cur.getString(2));

SimpleDateFormat format = new SimpleDateFormat(DATE\_FORMAT);

try {

note.setPostdate(format.parse(cur.getString(3)));

} catch (ParseException e) {

e.printStackTrace();

}

note.setPhoto(cur.getBlob(4));

int i = cur.getInt(5);

note.hasImage = i != 0;

note.setChildrenId(cur.getInt(6));

list.add(note);

}

return list;

}

public ArrayList<Note> getNotesByDate(Date date, int childrenId) {

SimpleDateFormat format = new SimpleDateFormat(DATE\_TIME\_FORMAT);

String dt = format.format(date);

String sql = "select \* from "

+ BabyProgressDataBaseHelper.NOTE\_TABLE\_NAME + " where "

+ BabyProgressDataBaseHelper.NOTE\_POSTDATE + " = '" + dt

+ "' and " + BabyProgressDataBaseHelper.NOTE\_CHILDREN\_ID

+ " = " + childrenId;

Cursor cur = db.rawQuery(sql, new String[] {});

ArrayList<Note> list = new ArrayList<Note>();

while (cur.moveToNext()) {

Note note = new Note();

note.setId(cur.getInt(0));

note.setDescription(cur.getString(1));

note.setTitle(cur.getString(2));

try {

note.setPostdate(format.parse(cur.getString(3)));

} catch (ParseException e) {

e.printStackTrace();

}

note.setPhoto(cur.getBlob(4));

int i = cur.getInt(5);

note.hasImage = i != 0;

note.setChildrenId(cur.getInt(6));

list.add(note);

}

return list;

}

public int updateChildren(Children children) {

ContentValues cv = new ContentValues();

cv.put(BabyProgressDataBaseHelper.CHILDREN\_AWATAR, children.getAwatar());

SimpleDateFormat format = new SimpleDateFormat(DATE\_TIME\_FORMAT);

String result = format.format(children.getBirthdate());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_BIRTHDATE, result);

cv.put(BabyProgressDataBaseHelper.CHILDREN\_GROWTH, children.getGrowth());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_MIDDLENAME,

children.getMiddlename());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_NAME, children.getName());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_PARENT\_ID,

children.getParentId());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_SURNAME,

children.getSurname());

cv.put(BabyProgressDataBaseHelper.CHILDREN\_WEIGHT, children.getWeight());

return db.update(BabyProgressDataBaseHelper.CHILDREN\_TABLE\_NAME, cv,

BabyProgressDataBaseHelper.CHILDREN\_ID + " = " + children.getId(), null);

}

public int updateNote(Note note) {

ContentValues cv = new ContentValues();

cv.put(BabyProgressDataBaseHelper.NOTE\_CHILDREN\_ID,

note.getChildrenId());

SimpleDateFormat format = new SimpleDateFormat(DATE\_TIME\_FORMAT);

String dRes = format.format(note.getPostdate());

cv.put(BabyProgressDataBaseHelper.NOTE\_POSTDATE, dRes);

cv.put(BabyProgressDataBaseHelper.NOTE\_TITLE, note.getTitle());

cv.put(BabyProgressDataBaseHelper.NOTE\_DESCRIPTION,

note.getDescription());

cv.put(BabyProgressDataBaseHelper.NOTE\_PHOTO, note.getPhoto());

int i = 0;

if (note.hasImage)

i = 1;

cv.put(BabyProgressDataBaseHelper.NOTE\_WITH\_IMAGE, i);

return db.update(BabyProgressDataBaseHelper.NOTE\_TABLE\_NAME, cv,

BabyProgressDataBaseHelper.NOTE\_ID + " = "+ note.getId(), null);

}

public int updateParent(Account parent) {

ContentValues cv = new ContentValues();

cv.put(BabyProgressDataBaseHelper.ACCOUNT\_PARENT\_BIRTHDATE, parent

.getBirthdate().toString());

cv.put(BabyProgressDataBaseHelper.ACCOUNT\_PARENT\_MIDDLENAME,

parent.getMiddlename());

cv.put(BabyProgressDataBaseHelper.ACCOUNT\_PARENT\_NAME, parent.getName());

cv.put(BabyProgressDataBaseHelper.ACCOUNT\_PARENT\_SURNAME,

parent.getSurname());

return db.update(BabyProgressDataBaseHelper.ACCOUNT\_TABLE\_NAME, cv,

BabyProgressDataBaseHelper.ACCOUNT\_ID + " = " + parent.getId(),

null);

}

public int updateNotice(Notice notice) {

ContentValues cv = new ContentValues();

cv.put(BabyProgressDataBaseHelper.NOTICE\_CHILDREN\_ID,

notice.getChildrenId());

SimpleDateFormat format = new SimpleDateFormat(

DataAdapter.DATE\_TIME\_FORMAT);

String dateRes = format.format(notice.getNotifyDateTime());

cv.put(BabyProgressDataBaseHelper.NOTICE\_NOTIFYDATETIME, dateRes);

cv.put(BabyProgressDataBaseHelper.NOTICE\_TITLE, notice.getTitle());

cv.put(BabyProgressDataBaseHelper.NOTICE\_DESCRIPTION,

notice.getDescription());

return db.update(BabyProgressDataBaseHelper.NOTICE\_TABLE\_NAME, cv,

BabyProgressDataBaseHelper.NOTICE\_ID + " = " + notice.getId(),

null);

}

public int deleteNote(Note note) {

return db.delete(BabyProgressDataBaseHelper.NOTE\_TABLE\_NAME,

BabyProgressDataBaseHelper.NOTE\_ID + " = "+ note.getId(), null);

}

public int deleteChildren(Children children) {

return db.delete(BabyProgressDataBaseHelper.CHILDREN\_TABLE\_NAME, BabyProgressDataBaseHelper.CHILDREN\_ID + " = "+ children.getId(), null);

}

public int deleteParent(Account account) {

return db

.delete(BabyProgressDataBaseHelper.ACCOUNT\_TABLE\_NAME,

BabyProgressDataBaseHelper.ACCOUNT\_ID + " = "

+ account.getId(), null);

}

public int deleteNotice(Notice notice) {

return db.delete(BabyProgressDataBaseHelper.NOTICE\_TABLE\_NAME,

BabyProgressDataBaseHelper.NOTICE\_ID + " = " + notice.getId(),null);

}

public ArrayList<Children> getChildrensByAccount(Account account) {

String sql = "select \* from "

+ BabyProgressDataBaseHelper.CHILDREN\_TABLE\_NAME + " where "

+ BabyProgressDataBaseHelper.CHILDREN\_PARENT\_ID + " = "

+ account.getId();

Cursor cur = db.rawQuery(sql, new String[] {});

ArrayList<Children> list = new ArrayList<Children>();

while (cur.moveToNext()) {

Children child = new Children();

child.setId(cur.getInt(0));

child.setName(cur.getString(1));

child.setSurname(cur.getString(2));

child.setMiddlename(cur.getString(3));

child.setWeight(cur.getDouble(4));

child.setGrowth(cur.getDouble(5));

child.setAwatar(cur.getBlob(6));

SimpleDateFormat format = new SimpleDateFormat(DATE\_FORMAT);

try {

child.setBirthdate(format.parse(cur.getString(7)));

} catch (ParseException e) {

e.printStackTrace();

}

child.setParentId(cur.getInt(8));

list.add(child);

}

return list;

}

public Account getAccountByLogin(String login) {

String sql = " select \* from "

+ BabyProgressDataBaseHelper.ACCOUNT\_TABLE\_NAME + " where "

+ BabyProgressDataBaseHelper.ACCOUNT\_LOGIN + " = '" + login

+ "'";

Cursor cur = db.rawQuery(sql, new String[] {});

ArrayList<Account> list = new ArrayList<Account>();

while (cur.moveToNext()) {

Account parent = new Account();

parent.setId(cur.getInt(0));

parent.setName(cur.getString(1));

parent.setSurname(cur.getString(2));

parent.setMiddlename(cur.getString(3));

SimpleDateFormat format = new SimpleDateFormat(DATE\_TIME\_FORMAT);

try {

parent.setBirthdate(format.parse(cur.getString(4)));

} catch (ParseException e) {

e.printStackTrace();

}

parent.setLogin(cur.getString(5));

parent.setPassword(cur.getString(6));

list.add(parent);

}

if (list.size() > 0)

return list.get(0);

else

return new Account();

}

public int getMaxNoticeId() {

String sql = " select max(" + BabyProgressDataBaseHelper.NOTICE\_ID

+ ") from " + BabyProgressDataBaseHelper.NOTICE\_TABLE\_NAME;

Cursor cur = db.rawQuery(sql, new String[] {});

try {

if (cur.moveToFirst()) {

return cur.getInt(0);

}

} catch (Exception e) {

return 0;

}

return 0;

}

}

1. Код класса «DataManager»

package com.example.babyprogressmap;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.InputStream;

import java.util.Date;

import states.ChangeChildrenProfileState;

import states.ChildrenProfileState;

import states.EditNoteState;

import states.EditNoticeState;

import states.LoginState;

import states.NotesState;

import states.NotificationsState;

import states.RegistrationState;

import states.State;

import states.ViewChildrenProfileState;

import states.ViewNoteState;

import android.content.Context;

import android.content.Intent;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.os.Build;

import forms.ChangeChildrenProfile;

import forms.ChildrenProfile;

import forms.EditNote;

import forms.Login;

import forms.Notes;

import forms.Notifications;

import forms.Registration;

import forms.ViewChildrenProfile;

import forms.ViewNote;

public class DataManager {

private static Account \_account;

private static Children \_children;

private static Note \_note;

private static Notice \_notice;

private static ActivityEnum \_PreviousActivity;

public static String Extra\_PreviousActivity = "previousActivity";

public static String Extra\_isUpdate = "isUpdate";

public static int MIN\_ALARM\_REQUEST\_ID = 2013;

private static Date \_date = new Date();

private static State \_currentState;

private static ChangeChildrenProfileState \_ChangeChildrenProfileState = new ChangeChildrenProfileState();

private static ChildrenProfileState \_ChildrenProfileState = new ChildrenProfileState();

private static EditNoteState \_EditNoteState = new EditNoteState();

private static LoginState \_LoginState = new LoginState();

private static NotesState \_NotesState = new NotesState();

private static RegistrationState \_RegistrationState = new RegistrationState();

private static ViewChildrenProfileState \_ViewChildrenProfileState = new ViewChildrenProfileState();

private static ViewNoteState \_ViewNoteState = new ViewNoteState();

private static NotificationsState \_NotificationsState = new NotificationsState();

private static EditNoticeState \_EditNoticeState = new EditNoticeState();

public static void reset() {

\_account = new Account();

\_children = new Children();

\_note = new Note();

\_notice = new Notice();

\_date = new Date();

setPreviousActivity(ActivityEnum.Null);

\_currentState = \_LoginState;

}

public static boolean hasLogged() {

return !(\_account.getId() == -1);

}

public static Date getChoosedDate() {

return \_date;

}

public static void setChoosedDate(Date date) {

\_date = date;

}

public static void setNotice(Notice notice) {

\_notice = notice;

}

public static Notice getNotice() {

return \_notice;

}

public static void setAccount(Account newAccount) {

\_account = newAccount;

}

// current account

public static Account getAccount() {

return \_account;

}

public static void setChildren(Children children) {

\_children = children;

}

// current children

public static Children getChildren() {

return \_children;

}

// current note

public static Note getNote() {

return \_note;

}

public static void setNote(Note newNote) {

\_note = newNote;

}

public static byte[] getImageBytes(Bitmap bitmap) {

try {

ByteArrayOutputStream stream = new ByteArrayOutputStream();

bitmap.compress(Bitmap.CompressFormat.PNG, 100, stream);

return stream.toByteArray();

} catch (Exception e) {

return new byte[0];

}

}

public static Bitmap getImageFromBytes(byte[] array) {

try {

InputStream is = new ByteArrayInputStream(array);

return BitmapFactory.decodeStream(is);

} catch (Exception e) {

Bitmap.Config conf = Bitmap.Config.ARGB\_8888;

return Bitmap.createBitmap(200, 200, conf);

}

}

public static Class<?> getClassByEnumValue(ActivityEnum value) {

switch (value) {

case ChangeChildrenProfile:

return ChangeChildrenProfile.class;

case EditNote:

return EditNote.class;

case ChildrenProfile:

return ChildrenProfile.class;

case Login:

return Login.class;

case Notes:

return Notes.class;

case Notifications:

return Notifications.class;

case Registration:

return Registration.class;

case ViewChildrenProfile:

return ViewChildrenProfile.class;

case ViewNote:

return ViewNote.class;

default:

return null;

}

}

public static int sizeOfBitmap(Bitmap data) {

if (Build.VERSION.SDK\_INT < Build.VERSION\_CODES.HONEYCOMB\_MR1) {

return data.getRowBytes() \* data.getHeight();

} else {

return data.getByteCount();

}

}

public Intent getNextFormIntent(Context context, ActivityEnum current,

ActivityEnum next) {

Intent intent = new Intent(context, getClassByEnumValue(next));

intent.putExtra(DataManager.Extra\_PreviousActivity, current.toString());

return intent;

}

public static ActivityEnum getPreviousActivity() {

return \_PreviousActivity;

}

public static void setPreviousActivity(ActivityEnum \_PreviousActivity) {

DataManager.\_PreviousActivity = \_PreviousActivity;

}

public static ChangeChildrenProfileState getChangeChildrenProfileState() {

return \_ChangeChildrenProfileState;

}

public static void setChangeChildrenProfileState(

ChangeChildrenProfileState \_ChangeChildrenProfileState) {

DataManager.\_ChangeChildrenProfileState = \_ChangeChildrenProfileState;

}

public static ChildrenProfileState getChildrenProfileState() {

return \_ChildrenProfileState;

}

public static void setChildrenProfileState(

ChildrenProfileState \_ChildrenProfileState) {

DataManager.\_ChildrenProfileState = \_ChildrenProfileState;

}

public static EditNoteState getEditNoteState() {

return \_EditNoteState;

}

public static void setEditNoteState(EditNoteState \_EditNoteState) {

DataManager.\_EditNoteState = \_EditNoteState;

}

public static LoginState getLoginState() {

return \_LoginState;

}

public static void setLoginState(LoginState \_LoginState) {

DataManager.\_LoginState = \_LoginState;

}

public static NotesState getNotesState() {

return \_NotesState;

}

public static void setNotesState(NotesState \_NotesState) {

DataManager.\_NotesState = \_NotesState;

}

public static RegistrationState getRegistrationState() {

return \_RegistrationState;

}

public static void setRegistrationState(RegistrationState \_RegistrationState) {

DataManager.\_RegistrationState = \_RegistrationState;

}

public static ViewChildrenProfileState getViewChildrenProfileState() {

return \_ViewChildrenProfileState;

}

public static void setViewChildrenProfileState(

ViewChildrenProfileState \_ViewChildrenProfileState) {

DataManager.\_ViewChildrenProfileState = \_ViewChildrenProfileState;

}

public static ViewNoteState getViewNoteState() {

return \_ViewNoteState;

}

public static void setViewNoteState(ViewNoteState \_ViewNoteState) {

DataManager.\_ViewNoteState = \_ViewNoteState;

}

public static State getCurrentState() {

return \_currentState;

}

public static void setCurrentState(State \_currentState) {

DataManager.\_currentState = \_currentState;

}

public static NotificationsState getNotificationsState() {

return \_NotificationsState;

}

public static void setNotificationsState(

NotificationsState \_NotificationsState) {

DataManager.\_NotificationsState = \_NotificationsState;

}

public static EditNoticeState getEditNoticeState() {

return \_EditNoticeState;

}

public static void setEditNoticeState(EditNoticeState \_EditNoticeState) {

DataManager.\_EditNoticeState = \_EditNoticeState;

}

}

1. Код класса «ListViewChildrenAdapter»

package com.example.babyprogressmap;

import java.util.ArrayList;

import android.content.Context;

import android.graphics.Color;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.ImageView;

import android.widget.TextView;

public class ListViewChildrenAdapter extends BaseAdapter {

private LayoutInflater mInflater = null;

private ArrayList<Children> childrens;

private final class ViewHolder {

ImageView imageView\_awatar;

TextView textView\_name;

}

private int selectedChildrenId = 0;

private int selectedPosition = 0;

private ViewHolder mHolder = null;

public ListViewChildrenAdapter(Context context) {

Context mContext = context;

mInflater = (LayoutInflater) mContext

.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

childrens = new ArrayList<Children>();

}

public int getSelectedPosition() {

return selectedPosition;

}

public void setSelectedPosition(int newPos) {

selectedPosition = newPos;

selectedChildrenId = childrens.get(newPos).getId();

}

public int getSelectedChildrenId() {

return selectedChildrenId;

}

public void addItem(Children child) {

childrens.add(child);

}

@Override

public int getCount() {

return childrens.size();

}

@Override

public Object getItem(int position) {

return childrens.get(position);

}

@Override

public long getItemId(int position) {

return childrens.get(position).getId();

}

public void clear()

{

childrens.clear();

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

if (convertView == null) {

mHolder = new ViewHolder();

convertView = mInflater.inflate(R.layout.list\_item\_children, null);

convertView.setTag(mHolder);

} else {

mHolder = (ViewHolder) convertView.getTag();

}

mHolder.imageView\_awatar = (ImageView) convertView

.findViewById(R.id.imageView\_awatar);

mHolder.imageView\_awatar.setImageBitmap(DataManager

.getImageFromBytes(childrens.get(position).getAwatar()));

mHolder.textView\_name = (TextView) convertView

.findViewById(R.id.textView\_name);

mHolder.textView\_name.setText(childrens.get(position).getName());

if (selectedPosition == position) {

convertView.setSelected(true);

convertView.setPressed(true);

convertView.setBackgroundColor(Color.GREEN);

selectedChildrenId = childrens.get(position).getId();

} else {

convertView.setSelected(false);

convertView.setPressed(false);

convertView.setBackgroundColor(Color.WHITE);

}

return convertView;

}

public void forceReload() {

notifyDataSetChanged();

}

}

1. Код класса «ListViewNoteAdapter»

package com.example.babyprogressmap;

import java.util.ArrayList;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.ImageView;

import android.widget.TextView;

public class ListViewNoteAdapter extends BaseAdapter {

private LayoutInflater mInflater = null;

private ArrayList<Note> notes = new ArrayList<Note>();

private final class ViewHolder {

ImageView imageView;

TextView textView\_tittle;

TextView textView\_description;

}

private ViewHolder mHolder = null;

public ListViewNoteAdapter(Context context) {

Context mContext = context;

mInflater = (LayoutInflater) mContext

.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

// notes = new ArrayList<Note>();

}

public ArrayList<Note> getCollection() {

return notes;

}

public void addItem(Note note) {

notes.add(note);

}

@Override

public int getCount() {

return notes.size();

}

@Override

public Object getItem(int position) {

return notes.get(position);

}

@Override

public long getItemId(int position) {

return notes.get(position).getId();

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

boolean hasImage = notes.get(position).hasImage;

mHolder = new ViewHolder();

if (hasImage)

convertView = mInflater.inflate(R.layout.list\_item\_note, null);

else {

convertView = mInflater.inflate(

R.layout.list\_item\_note\_without\_image, null);

}

convertView.setTag(mHolder);

if (hasImage) {

mHolder.imageView = (ImageView) convertView.findViewById(R.id.imageView);

mHolder.imageView.setImageBitmap(DataManager .getImageFromBytes(notes.get(position).getPhoto()));

}

mHolder.textView\_tittle = (TextView) convertView

.findViewById(R.id.textView\_title);

mHolder.textView\_tittle.setText(notes.get(position).getTitle());

mHolder.textView\_description = (TextView) convertView

.findViewById(R.id.textView\_description);

mHolder.textView\_description.setText(notes.get(position)

.getDescription());

return convertView;

}

public void forceReload() {

notifyDataSetChanged();

}

}

1. Код класс «ListViewNoticeAdapter»

package com.example.babyprogressmap;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import android.content.Context;

import android.graphics.Color;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.ImageView;

import android.widget.TextView;

public class ListViewNoticeAdapter extends BaseAdapter {

private LayoutInflater mInflater = null;

private ArrayList<Notice> notifications;

private final class ViewHolder {

TextView textView\_datetime;

TextView textView\_title;

TextView textView\_description;

}

private int selectedChildrenId = 0;

private int selectedPosition = 0;

private ViewHolder mHolder = null;

public ListViewNoticeAdapter(Context context) {

Context mContext = context;

mInflater = (LayoutInflater) mContext

.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

notifications = new ArrayList<Notice>();

}

public int getSelectedPosition() {

return selectedPosition;

}

public void setSelectedPosition(int newPos) {

selectedPosition = newPos;

selectedChildrenId = notifications.get(newPos).getId();

}

public int getSelectedChildrenId() {

return selectedChildrenId;

}

public void addItem(Notice notice) {

notifications.add(notice);

}

@Override

public int getCount() {

return notifications.size();

}

@Override

public Object getItem(int position) {

return notifications.get(position);

}

@Override

public long getItemId(int position) {

return notifications.get(position).getId();

}

public ArrayList<Notice> getCollection()

{

return notifications;

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

if (convertView == null) {

mHolder = new ViewHolder();

convertView = mInflater.inflate(R.layout.list\_item\_notice, null);

convertView.setTag(mHolder);

} else {

mHolder = (ViewHolder) convertView.getTag();

}

mHolder.textView\_title = (TextView) convertView

.findViewById(R.id.textView\_title);

mHolder.textView\_title.setText(notifications.get(position).getTitle());

mHolder.textView\_description = (TextView) convertView

.findViewById(R.id.textView\_description);

mHolder.textView\_description.setText(notifications.get(position).getDescription());

SimpleDateFormat format = new SimpleDateFormat(DataAdapter.DATE\_TIME\_FORMAT\_REVERSED);

String dt = format.format(notifications.get(position).getNotifyDateTime());

mHolder.textView\_datetime = (TextView) convertView

.findViewById(R.id.textView\_datetime);

mHolder.textView\_datetime.setText(dt);

if (selectedPosition == position) {

selectedChildrenId = notifications.get(position).getId();

}

public void forceReload() {

notifyDataSetChanged();

}

}

1. Код класса «MyAsyncTask»

package com.example.babyprogressmap;

import android.content.Context;

import android.os.AsyncTask;

import android.widget.Toast;

public class MyAsyncTask extends AsyncTask<String, Void, String> {

private Context context;

public MyAsyncTask(Context context) {

this.context = context;

}

private String message = "";

@Override

protected String doInBackground(String... params) {

this.message = params[0];

return "";

}

@Override

protected void onPostExecute(String result) {

super.onPostExecute(result);

Toast.makeText(context, message, Toast.LENGTH\_LONG).show();

}

@Override

protected void onPreExecute() {

super.onPreExecute();

}

}

1. Код класса «Note»

package com.example.babyprogressmap;

import java.util.Date;

public class Note {

public Note() {

\_Id = -1;

\_ChildrenId = -1;

\_Description = "";

\_Photo = null;

\_Postdate = new Date();

\_withImage = true;

}

private int \_Id;

private int \_ChildrenId;

private String \_Description;

private byte[] \_Photo;

private Date \_Postdate;

private String \_Title;

private boolean \_withImage;

public boolean hasImage = true; // TODO

public void setId(int newId) {

\_Id = newId;

}

public int getId() {

return \_Id;

}

public void setChildrenId(int newChildrenId) {

\_ChildrenId = newChildrenId;

}

public int getChildrenId() {

return \_ChildrenId;

}

public void setDescription(String newDescription) {

\_Description = newDescription;

}

public String getDescription() {

return \_Description;

}

public void setPhoto(byte[] newPhoto) {

\_Photo = newPhoto;

}

public byte[] getPhoto() {

return \_Photo;

}

public void setPostdate(Date newPostdate) {

\_Postdate = newPostdate;

}

public Date getPostdate() {

return \_Postdate;

}

public String getTitle() {

return \_Title;

}

public void setTitle(String newTitle) {

\_Title = newTitle;

}

public String toString() {

String result = "\_Postdate = " + \_Postdate.toLocaleString()

+ "\n\_ChildrenId = " + \_ChildrenId + "\n\_Description = "

+ \_Description + "\n\_Id = " + \_Id + "\n\_Title = " + \_Title

+ "\n\_withImage = " + \_withImage;

return result;

}

1. Код класса «Notice»

package com.example.babyprogressmap;

import java.util.Date;

public class Notice {

public Notice() {

\_Id = -1;

\_Title = "";

\_Description = "";

\_NotifyDateTime = new Date();

\_ChildrenId = -1;

}

private int \_Id;

private String \_Title;

private String \_Description;

private Date \_NotifyDateTime;

private int \_ChildrenId;

public void setId(int newId) {

\_Id = newId;

}

public int getId() {

return \_Id;

}

public int getChildrenId() {

return \_ChildrenId;

}

public void setChildrenId(int newChildrenId) {

\_ChildrenId = newChildrenId;

}

public String getTitle() {

return \_Title;

}

public void setTitle(String newTitle) {

\_Title = newTitle;

}

public String getDescription() {

return \_Description;

}

public void setDescription(String newDescription) {

\_Description = newDescription;

}

public void setNotifyDateTime(Date newNotifyDateTime) {

\_NotifyDateTime = newNotifyDateTime;

}

public Date getNotifyDateTime() {

return \_NotifyDateTime;

}

}

1. Код класса «NotificationAsyncTask»

package com.example.babyprogressmap;

import android.content.Context;

import android.os.AsyncTask;

import android.widget.Toast;

public class NotificationAsyncTask extends AsyncTask<String, Void, String> {

private Context context;

public NotificationAsyncTask(Context context) {

this.context = context;

}

/\*\*

\* @param 0 = title

\* @param 1 = description

\* \*/

@Override

protected String doInBackground(String... params) {

NotificationUtils.getInstance(context).createInfoNotification(

params[0], params[1]);

return "";

}

@Override

protected void onPostExecute(String result) {

super.onPostExecute(result);

}

@Override

protected void onPreExecute() {

super.onPreExecute();

}

}

1. Код класса «NotificationUtils»

package com.example.babyprogressmap;

import java.util.HashMap;

import android.app.Notification;

import android.app.NotificationManager;

import android.app.PendingIntent;

import android.content.Context;

import android.content.Intent;

import android.support.v4.app.NotificationCompat;

import forms.Notifications;

public class NotificationUtils {

private static final String TAG = NotificationUtils.class.getSimpleName();

private static NotificationUtils instance;

private static Context context;

private NotificationManager manager;

private int lastId = 0;

private HashMap<Integer, Notification> notifications;

private NotificationUtils(Context context) {

this.context = context;

manager = (NotificationManager) context

.getSystemService(Context.NOTIFICATION\_SERVICE);

notifications = new HashMap<Integer, Notification>();

}

public static NotificationUtils getInstance(Context context) {

if (instance == null) {

instance = new NotificationUtils(context);

} else {

instance.context = context;

}

return instance;

}

public int createInfoNotification(String title, String message) {

Intent notificationIntent = new Intent(context, Notifications.class);

NotificationCompat.Builder nb = new NotificationCompat.Builder(context)

.setSmallIcon(R.drawable.launcher\_icon)

.setAutoCancel(true)

.setTicker(title)

.setContentText(message)

.setContentIntent(

PendingIntent.getActivity(context, 0,

notificationIntent,

PendingIntent.FLAG\_CANCEL\_CURRENT))

.setWhen(System.currentTimeMillis()).setContentTitle(title)

.setDefaults(Notification.DEFAULT\_ALL);

Notification notification = nb.getNotification();

manager.notify(lastId, notification);

notifications.put(lastId, notification);

return lastId++;

}

}

#### ПРИЛОЖЕНИЕ Б

Код классов из пакета «forms»

1. Код класса «ChangeChildrenProfile»

package forms;

import java.lang.reflect.Field;

import java.util.ArrayList;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.Children;

import com.example.babyprogressmap.DataAdapter;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.ListViewChildrenAdapter;

import com.example.babyprogressmap.MyAsyncTask;

import com.example.babyprogressmap.R;

import com.example.babyprogressmap.R.id;

import com.example.babyprogressmap.R.layout;

import com.example.babyprogressmap.R.menu;

import android.annotation.SuppressLint;

import android.app.ActionBar;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.graphics.Bitmap;

import android.graphics.Color;

import android.graphics.drawable.Drawable;

import android.os.Bundle;

import android.view.ContextMenu;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewConfiguration;

import android.view.ContextMenu.ContextMenuInfo;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.AdapterView.OnItemLongClickListener;

import android.widget.Button;

import android.widget.ImageButton;

import android.widget.ImageView;

import android.widget.ListView;

import android.widget.Toast;

public class ChangeChildrenProfile extends Activity {

ImageButton imageButton\_add;

DataAdapter dataAdapter;

ListViewChildrenAdapter listViewAdapter;

ListView listView\_children;

View row;

Bundle bundle;

int selectedChildrenId = -1;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

bundle = savedInstanceState;

setContentView(R.layout.activity\_change\_children\_profile);

DataManager.getCurrentState().setActivity(this);

imageButton\_add = (ImageButton) findViewById(R.id.imageButton\_add);

listView\_children = (ListView) findViewById(R.id.listView\_children);

dataAdapter = new DataAdapter(this);

dataAdapter.open();

loadAdapterCollection();

listView\_children

.setOnItemLongClickListener(new OnItemLongClickListener() {

public boolean onItemLongClick(AdapterView<?> a, View v,

int position, long id) {

chooseItem(position);

return false;

}

});

listView\_children.setOnItemClickListener(new OnItemClickListener() {

public void onItemClick(AdapterView<?> a, View v, int position,

long id) {

chooseItem(position);

}

});

listView\_children.setLongClickable(true);

registerForContextMenu(listView\_children);

imageButton\_add.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View arg0) {

// Intent myIntent = new Intent(getBaseContext(),

// ChildrenProfile.class);

// myIntent.putExtra("isUpdate", false);

// myIntent.putExtra("previousActivity", "ChangeChildrenProfile");

// startActivity(myIntent);

DataManager.getCurrentState().middleButtonBarButtonClicked(getBaseContext());

}

});

ActionBar aBar = getActionBar();

aBar.setDisplayShowTitleEnabled(false);

aBar.setDisplayShowHomeEnabled(false);

aBar.show();

getOverflowMenu();

switch (DataManager.getPreviousActivity()) {

case ChildrenProfile:

if (listViewAdapter.getCount() > 0)

chooseItem(listViewAdapter.getCount() - 1);

break;

}

}

private void chooseItem(int position) {

listViewAdapter.setSelectedPosition(position);

selectedChildrenId = listViewAdapter.getSelectedChildrenId();

listView\_children.setAdapter(listViewAdapter);

ArrayList<Children> childrensList = dataAdapter.getChildrens();

for (Children ch : childrensList) {

if (ch.getId() == selectedChildrenId) {

DataManager.setChildren(ch);

}

}

}

private void getOverflowMenu() {

try {

ViewConfiguration config = ViewConfiguration.get(this);

Field menuKeyField = ViewConfiguration.class

.getDeclaredField("sHasPermanentMenuKey");

if (menuKeyField != null) {

menuKeyField.setAccessible(true);

menuKeyField.setBoolean(config, false);

}

} catch (Exception e) {

e.printStackTrace();

}

}

public void onCreateContextMenu(ContextMenu menu, View v,

ContextMenuInfo menuInfo) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.context\_menu\_change\_children\_profile, menu);

super.onCreateContextMenu(menu, v, menuInfo);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.action\_bar\_change\_children\_profile, menu);

return true;

}

@SuppressLint("ShowToast")

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (item.getItemId() != R.id.logout) {

int count = listViewAdapter.getCount();

if (count == 0) {

new MyAsyncTask(getApplicationContext()).execute("Нет ни одного профиля");

return true;

} else if (selectedChildrenId == -1) {

new MyAsyncTask(getApplicationContext()).execute("Ничего не выбрано");

return true;

}

}

switch (item.getItemId()) {

case R.id.children\_profile:

DataManager.getCurrentState().viewChildrenProfileClicked(getBaseContext());

return true;

case R.id.notes:

DataManager.getCurrentState().notesClicked(getBaseContext());

return true;

case R.id.notifications:

DataManager.getCurrentState().notificationsClicked(getBaseContext());

return true;

case R.id.logout:

DataManager.getCurrentState().logoutClicked(getBaseContext());

return true;

default:

return super.onOptionsItemSelected(item);

}

}

@SuppressWarnings("deprecation")

public boolean onContextItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.edit:

DataManager.getCurrentState().childrenProfileClicked(getBaseContext());

return true;

case R.id.delete:

AlertDialog alert = new AlertDialog.Builder(this).create();

alert.setTitle("");

alert.setMessage("Вы действительно хотите удалить выбранный профиль?");

alert.setButton("Да", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

deleteChoosedItem();

}

});

alert.setButton2("Нет", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

// ...

}

});

alert.show();

return true;

}

return super.onContextItemSelected(item);

}

private void deleteChoosedItem() {

dataAdapter.deleteChildren(DataManager.getChildren());

loadAdapterCollection();

}

private void loadAdapterCollection() {

ArrayList<Children> childrens = dataAdapter

.getChildrensByAccount(DataManager.getAccount());

listViewAdapter = new ListViewChildrenAdapter(this);

int selected = -1;

for (int i = 0; i < childrens.size(); i++) {

listViewAdapter.addItem(childrens.get(i));

if (childrens.get(i).getId() == DataManager.getChildren().getId())

selected = i;

}

if (listViewAdapter.getCount() > 0) {

switch (selected) {

case -1:

chooseItem(0);

break;

default:

chooseItem(selected);

}

}

listView\_children.setAdapter(listViewAdapter);

listViewAdapter.notifyDataSetChanged();

}

@Override

public void onBackPressed() {

}

}

1. Код класса «ChildrenProfile»

package forms;

import java.io.ByteArrayOutputStream;

import java.io.FileNotFoundException;

import java.io.InputStream;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.Children;

import com.example.babyprogressmap.DataAdapter;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.R;

import com.example.babyprogressmap.R.id;

import com.example.babyprogressmap.R.layout;

import com.example.babyprogressmap.R.menu;

import com.example.babyprogressmap.R.string;

import android.animation.Animator;

import android.animation.AnimatorListenerAdapter;

import android.annotation.TargetApi;

import android.app.Activity;

import android.content.Intent;

import android.content.res.Configuration;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.drawable.BitmapDrawable;

import android.net.Uri;

import android.os.AsyncTask;

import android.os.Build;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.ContextMenu;

import android.view.KeyEvent;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.ContextMenu.ContextMenuInfo;

import android.view.inputmethod.EditorInfo;

import android.widget.Button;

import android.widget.DatePicker;

import android.widget.EditText;

import android.widget.ImageButton;

import android.widget.ImageView;

import android.widget.TextView;

public class ChildrenProfile extends Activity {

ImageView imageView\_Awatar;

ImageButton imageButton\_options;

EditText edit\_name;

EditText edit\_surname;

EditText edit\_middlename;

EditText edit\_weight;

EditText edit\_growth;

DatePicker datepicker\_birthdate;

Button button\_ok;

Button button\_cancel;

int index = 0;

int parentId = -1;

Bitmap selectedBitmap = null;

Intent intent;

private static final int SELECT\_PHOTO = 100;

private static final int CAMERA\_REQUEST = 1888;

boolean isUpdate;

DataAdapter adapter;

@SuppressWarnings("deprecation")

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_children\_profile\_hints);

DataManager.getCurrentState().setActivity(this);

imageButton\_options = (ImageButton) findViewById(R.id.imageButton\_options);

imageView\_Awatar = (ImageView) findViewById(R.id.imageView\_Awatar);

edit\_name = (EditText) findViewById(R.id.edit\_name);

edit\_surname = (EditText) findViewById(R.id.edit\_surname);

edit\_middlename = (EditText) findViewById(R.id.edit\_middlename);

edit\_weight = (EditText) findViewById(R.id.edit\_weight);

edit\_growth = (EditText) findViewById(R.id.edit\_growth);

datepicker\_birthdate = (DatePicker) findViewById(R.id.datePicker\_birthdate);

button\_ok = (Button) findViewById(R.id.button\_ok);

button\_cancel = (Button) findViewById(R.id.button\_cancel);

adapter = new DataAdapter(this);

adapter.open();

intent = getIntent();

isUpdate = intent.getBooleanExtra("isUpdate", false);

parentId = DataManager.getAccount().getId();

if (isUpdate) {

edit\_name.setText(DataManager.getChildren().getName());

edit\_surname.setText(DataManager.getChildren().getSurname());

edit\_middlename.setText(DataManager.getChildren().getMiddlename());

edit\_weight.setText(DataManager.getChildren().getWeight() + "");

edit\_growth.setText(DataManager.getChildren().getGrowth() + "");

index = DataManager.getChildren().getId();

Date date = DataManager.getChildren().getBirthdate();

Calendar calendar = Calendar.getInstance();

calendar.setTime(date);

datepicker\_birthdate.init(calendar.get(Calendar.YEAR),

calendar.get(Calendar.MONTH),

calendar.get(Calendar.DAY\_OF\_MONTH), null);

// datepicker\_birthdate.updateDate(date.getYear() - 1900,

// date.getMonth()-1, date.getDay());

imageView\_Awatar.setImageBitmap(DataManager

.getImageFromBytes(DataManager.getChildren().getAwatar()));

}

button\_ok.setOnClickListener(new View.OnClickListener() {

@SuppressWarnings("deprecation")

@Override

public void onClick(View v) {

Children children = new Children();

children.setId(index);

Bitmap bitmap = ((BitmapDrawable) imageView\_Awatar

.getDrawable()).getBitmap();

children.setAwatar(DataManager.getImageBytes(bitmap)); // TODO

Date date = new Date(datepicker\_birthdate.getYear() - 1900,

datepicker\_birthdate.getMonth(),

datepicker\_birthdate.getDayOfMonth());

children.setBirthdate(date);

try {

children.setGrowth(Double.parseDouble(edit\_growth.getText()

.toString()));

} catch (NumberFormatException e) {

edit\_growth.setError(getString(R.string.wrong\_value));

return;

}

try {

children.setWeight(Double.parseDouble(edit\_weight.getText()

.toString()));

} catch (NumberFormatException e) {

edit\_weight.setError(getString(R.string.wrong\_value));

return;

}

children.setMiddlename(edit\_middlename.getText().toString());

children.setName(edit\_name.getText().toString());

children.setSurname(edit\_surname.getText().toString());

children.setParentId(parentId);

if (isUpdate)

adapter.updateChildren(children);

else

adapter.insertChildren(children);

DataManager.setChildren(children);

DataManager.getCurrentState().leftButtonBarButtonClicked(getBaseContext());

}

});

imageButton\_options.setLongClickable(false);

registerForContextMenu(imageButton\_options);

imageButton\_options.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

openContextMenu(v);

}

});

button\_cancel.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

DataManager.getCurrentState().rightButtonBarButtonClicked(getBaseContext());

}

});

}

public void onCreateContextMenu(ContextMenu menu, View v,

ContextMenuInfo menuInfo) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.context\_menu\_children\_profile\_edit\_image, menu);

super.onCreateContextMenu(menu, v, menuInfo);

}

@Override

public boolean onContextItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.from\_camera:

Intent cameraIntent = new Intent(

android.provider.MediaStore.ACTION\_IMAGE\_CAPTURE);

startActivityForResult(cameraIntent, CAMERA\_REQUEST);

imageView\_Awatar.setVisibility(View.VISIBLE);

return true;

case R.id.from\_gallery:

Intent photoPickerIntent = new Intent(Intent.ACTION\_PICK);

photoPickerIntent.setType("image/\*");

startActivityForResult(photoPickerIntent, SELECT\_PHOTO);

imageView\_Awatar.setVisibility(View.VISIBLE);

return true;

default:

return true;

}

}

@Override

protected void onActivityResult(int requestCode, int resultCode,

Intent imageReturnedIntent) {

super.onActivityResult(requestCode, resultCode, imageReturnedIntent);

switch (requestCode) {

case CAMERA\_REQUEST:

if (resultCode == RESULT\_OK) {

selectedBitmap = (Bitmap) imageReturnedIntent.getExtras().get(

"data");

}

break;

case SELECT\_PHOTO:

if (resultCode == RESULT\_OK) {

Uri selectedImage = imageReturnedIntent.getData();

InputStream imageStream = null;

try {

imageStream = getContentResolver().openInputStream(

selectedImage);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

selectedBitmap = BitmapFactory.decodeStream(imageStream);

}

break;

}

if (resultCode == RESULT\_OK) {

imageView\_Awatar.setImageBitmap(selectedBitmap);

imageView\_Awatar.refreshDrawableState();

}

}

@Override

public void onConfigurationChanged(Configuration newConfig) {

super.onConfigurationChanged(newConfig);

}

@Override

public void onBackPressed() {

}

}

1. Код класса «EditNote»

package forms;

import java.io.FileNotFoundException;

import java.io.InputStream;

import java.util.Calendar;

import java.util.Date;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataAdapter;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.MyAsyncTask;

import com.example.babyprogressmap.Note;

import com.example.babyprogressmap.R;

import com.example.babyprogressmap.R.id;

import com.example.babyprogressmap.R.layout;

import com.example.babyprogressmap.R.menu;

import android.app.Activity;

import android.content.Intent;

import android.content.res.Configuration;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.drawable.BitmapDrawable;

import android.net.Uri;

import android.os.Bundle;

import android.view.ContextMenu;

import android.view.ContextMenu.ContextMenuInfo;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.DatePicker;

import android.widget.EditText;

import android.widget.ImageButton;

import android.widget.ImageView;

public class EditNote extends Activity {

private static final int SELECT\_PHOTO = 100;

private static final int CAMERA\_REQUEST = 1888;

Bitmap selectedBitmap = null;

DataAdapter adapter;

boolean hasImage = true;

ImageView imageView\_photo;

ImageButton imageButton\_options;

EditText editText\_title;

EditText editText\_description;

Button button\_ok;

Button button\_cancel;

DatePicker datePicker\_postdate;

Intent intent;

boolean isUpdate;

boolean withoutImageChecked = false;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_edit\_note);

DataManager.getCurrentState().setActivity(this);

adapter = new DataAdapter(this);

adapter.open();

imageView\_photo = (ImageView) findViewById(R.id.imageView\_photo);

imageButton\_options = (ImageButton) findViewById(R.id.imageButton\_options);

editText\_title = (EditText) findViewById(R.id.editText\_title);

editText\_description = (EditText) findViewById(R.id.editText\_description);

button\_ok = (Button) findViewById(R.id.button\_ok);

button\_cancel = (Button) findViewById(R.id.button\_cancel);

datePicker\_postdate = (DatePicker) findViewById(R.id.datePicker\_postdate);

imageButton\_options.setLongClickable(false);

registerForContextMenu(imageButton\_options);

imageButton\_options.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

openContextMenu(v);

}

});

intent = getIntent();

isUpdate = intent.getBooleanExtra("isUpdate", false);

if (isUpdate) {

Note currentNote = DataManager.getNote();

hasImage = currentNote.hasImage;

if (currentNote.hasImage)

imageView\_photo.setImageBitmap(DataManager

.getImageFromBytes(currentNote.getPhoto()));

else

imageView\_photo.setVisibility(View.GONE);

Date date = DataManager.getNote().getPostdate();

Calendar calendar = Calendar.getInstance();

calendar.setTime(date);

datePicker\_postdate.init(calendar.get(Calendar.YEAR),

calendar.get(Calendar.MONTH),

calendar.get(Calendar.DAY\_OF\_MONTH), null);

editText\_description.setText(currentNote.getDescription());

editText\_title.setText(currentNote.getTitle());

} else {

Date date = DataManager.getChoosedDate();

Calendar calendar = Calendar.getInstance();

calendar.setTime(date);

datePicker\_postdate.init(calendar.get(Calendar.YEAR),

calendar.get(Calendar.MONTH),

calendar.get(Calendar.DAY\_OF\_MONTH), null);

}

button\_ok.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Note note = DataManager.getNote();

@SuppressWarnings("deprecation")

Date date = new Date(datePicker\_postdate.getYear() - 1900,

datePicker\_postdate.getMonth(), datePicker\_postdate

.getDayOfMonth());

note.setPostdate(date);

note.setChildrenId(DataManager.getChildren().getId());

note.setDescription(editText\_description.getText().toString());

note.setTitle(editText\_title.getText().toString());

note.hasImage = hasImage;

if (hasImage) {

Bitmap bitmap = ((BitmapDrawable) imageView\_photo

.getDrawable()).getBitmap();

note.setPhoto(DataManager.getImageBytes(bitmap));

}

if (isUpdate) {

adapter.updateNote(note);

} else {

adapter.insertNote(note);

}

DataManager.setNote(note);

DataManager.getCurrentState().leftButtonBarButtonClicked(

getBaseContext());

}

});

button\_cancel.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

DataManager.getCurrentState().rightButtonBarButtonClicked(

getBaseContext());

}

});

}

public void onCreateContextMenu(ContextMenu menu, View v,

ContextMenuInfo menuInfo) {

super.onCreateContextMenu(menu, v, menuInfo);

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.context\_menu\_edit\_image\_options, menu);

for (int i = 0; i < menu.size(); ++i) {

MenuItem mi = menu.getItem(i);

if (mi.getItemId() == R.id.without\_image) {

mi.setChecked(withoutImageChecked);

}

}

}

@Override

public boolean onContextItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.from\_camera:

Intent cameraIntent = new Intent(

android.provider.MediaStore.ACTION\_IMAGE\_CAPTURE);

startActivityForResult(cameraIntent, CAMERA\_REQUEST);

imageView\_photo.setVisibility(View.VISIBLE);

return true;

case R.id.from\_gallery:

Intent photoPickerIntent = new Intent(Intent.ACTION\_PICK);

photoPickerIntent.setType("image/\*");

startActivityForResult(photoPickerIntent, SELECT\_PHOTO);

imageView\_photo.setVisibility(View.VISIBLE);

return true;

case R.id.without\_image:

if (imageView\_photo.getVisibility() == View.GONE)

imageView\_photo.setVisibility(View.VISIBLE);

else

imageView\_photo.setVisibility(View.GONE);

withoutImageChecked = !withoutImageChecked;

hasImage = !hasImage;

return true;

default:

return true;

}

}

@Override

protected void onActivityResult(int requestCode, int resultCode,

Intent imageReturnedIntent) {

super.onActivityResult(requestCode, resultCode, imageReturnedIntent);

switch (requestCode) {

case CAMERA\_REQUEST:

if (resultCode == RESULT\_OK) {

selectedBitmap = (Bitmap) imageReturnedIntent.getExtras().get(

"data");

}

break;

case SELECT\_PHOTO:

if (resultCode == RESULT\_OK) {

Uri selectedImage = imageReturnedIntent.getData();

InputStream imageStream = null;

try {

imageStream = getContentResolver().openInputStream(

selectedImage);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

selectedBitmap = BitmapFactory.decodeStream(imageStream);

if (selectedBitmap.getHeight() > 2048

| selectedBitmap.getWidth() > 2048) {

new MyAsyncTask(getBaseContext())

.execute("Слишком большой размер изображения");

resultCode = Activity.RESULT\_CANCELED;

}

}

break;

}

if (resultCode == RESULT\_OK) {

imageView\_photo.setImageBitmap(selectedBitmap);

imageView\_photo.refreshDrawableState();

}

}

@Override

public void onConfigurationChanged(Configuration newConfig) {

super.onConfigurationChanged(newConfig);

}

@Override

public void onBackPressed() {

}

}

1. Код класса «EditNotice»

package forms;

import java.util.Calendar;

import java.util.Date;

import android.app.Activity;

import android.app.AlarmManager;

import android.app.PendingIntent;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.DatePicker;

import android.widget.EditText;

import android.widget.TimePicker;

import com.example.babyprogressmap.AlarmReceiver;

import com.example.babyprogressmap.DataAdapter;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.MyAsyncTask;

import com.example.babyprogressmap.Notice;

import com.example.babyprogressmap.R;

public class EditNotice extends Activity {

EditText editText\_title;

EditText editText\_description;

DatePicker datePicker\_postdate;

TimePicker timePicker\_posttime;

Button button\_ok;

Button button\_back;

int index = -1;

DataAdapter dataAdapter;

boolean isUpdate = false;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_edit\_notice);

DataManager.getCurrentState().setActivity(this);

dataAdapter = new DataAdapter(this);

dataAdapter.open();

editText\_title = (EditText) findViewById(R.id.editText\_title);

editText\_description = (EditText) findViewById(R.id.editText\_description);

datePicker\_postdate = (DatePicker) findViewById(R.id.datePicker\_postdate);

timePicker\_posttime = (TimePicker) findViewById(R.id.timePicker\_posttime);

timePicker\_posttime.setIs24HourView(true);

button\_ok = (Button) findViewById(R.id.button\_ok);

button\_back = (Button) findViewById(R.id.button\_back);

Intent intent = getIntent();

isUpdate = intent.getBooleanExtra("isUpdate", false);

if (isUpdate) {

Notice notice = DataManager.getNotice();

index = notice.getId();

editText\_title.setText(notice.getTitle());

editText\_description.setText(notice.getDescription());

Date postdate = notice.getNotifyDateTime();

Calendar calendar = Calendar.getInstance();

calendar.setTime(postdate);

datePicker\_postdate.init(calendar.get(Calendar.YEAR),

calendar.get(Calendar.MONTH),

calendar.get(Calendar.DAY\_OF\_MONTH), null);

timePicker\_posttime.setCurrentHour(calendar

.get(Calendar.HOUR\_OF\_DAY));

timePicker\_posttime.setCurrentMinute(calendar.get(Calendar.MINUTE));

}

else

{

Calendar calendar = Calendar.getInstance();

timePicker\_posttime.setCurrentHour(calendar

.get(Calendar.HOUR\_OF\_DAY));

timePicker\_posttime.setCurrentMinute(calendar.get(Calendar.MINUTE));

}

button\_ok.setOnClickListener(new View.OnClickListener() {

@SuppressWarnings("deprecation")

@Override

public void onClick(View v) {

Notice notice = new Notice();

notice.setId(index);

notice.setDescription(editText\_description.getText().toString());

notice.setTitle(editText\_title.getText().toString());

Calendar cal = Calendar.getInstance();

cal.setFirstDayOfWeek(Calendar.MONDAY);

cal.set(Calendar.HOUR\_OF\_DAY, timePicker\_posttime.getCurrentHour());

cal.set(Calendar.MINUTE, timePicker\_posttime.getCurrentMinute());

cal.set(Calendar.YEAR, datePicker\_postdate.getYear());

cal.set(Calendar.MONTH, datePicker\_postdate.getMonth());

cal.set(Calendar.DAY\_OF\_MONTH,

datePicker\_postdate.getDayOfMonth());

Date date = cal.getTime();

// = new Date(datePicker\_postdate.getYear() - 1900,

// datePicker\_postdate.getMonth(), datePicker\_postdate

// .getDayOfMonth(), timePicker\_posttime

// .getCurrentHour(), timePicker\_posttime

// .getCurrentMinute());

notice.setNotifyDateTime(date);

notice.setChildrenId(DataManager.getChildren().getId());

DataManager.setNotice(notice);

if (isUpdate) {

dataAdapter.updateNotice(notice);

index = index + DataManager.MIN\_ALARM\_REQUEST\_ID;

} else {

long addedId = dataAdapter.insertNotice(notice);

index = DataManager.MIN\_ALARM\_REQUEST\_ID + (int) addedId;

}

// добавление уведомления с инициацией сообщения в статус баре

Intent intent = new Intent(getBaseContext(),

AlarmReceiver.class);

intent.putExtra("alarm\_message", notice.getDescription());

intent.putExtra("title", notice.getTitle());

PendingIntent sender = PendingIntent.getBroadcast(

getBaseContext(), index, intent,

PendingIntent.FLAG\_UPDATE\_CURRENT);

AlarmManager am = (AlarmManager) getSystemService(ALARM\_SERVICE);

am.set(AlarmManager.RTC\_WAKEUP, cal.getTimeInMillis(), sender);

DataManager.getCurrentState().leftButtonBarButtonClicked(

getBaseContext());

}

});

button\_back.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View arg0) {

DataManager.getCurrentState().rightButtonBarButtonClicked(

getBaseContext());

}

});

}

@Override

public void onBackPressed() {

}

}

1. Код класса «Login»

package forms;

import java.util.ArrayList;

import states.LoginState;

import com.example.babyprogressmap.Account;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.Children;

import com.example.babyprogressmap.DataAdapter;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.R;

import com.example.babyprogressmap.R.id;

import com.example.babyprogressmap.R.layout;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

public class Login extends Activity {

EditText email;

EditText password;

Button sign\_in\_button;

String login;

String pass;

DataAdapter adapter;

AlertDialog.Builder alert;

private final int \_passwordLength = 6;

boolean registered = false;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

DataManager.reset();

setContentView(R.layout.activity\_login);

DataManager.getCurrentState().setActivity(this);

email = (EditText) findViewById(R.id.email);

password = (EditText) findViewById(R.id.password);

adapter = new DataAdapter(this);

if (adapter.isClosed())

adapter.open();

findViewById(R.id.sign\_in\_button).setOnClickListener(

new View.OnClickListener() {

@Override

public void onClick(View view) {

if (adapter.isClosed())

adapter.open();

attemptLogin();

}

});

}

private void attemptLogin() {

boolean registered = false;

boolean verifiedPass = false;

login = email.getText().toString();

pass = password.getText().toString();

if (!isPasswordValid(pass)) {

password.setError("Пароль должен быть не меньше " + \_passwordLength

+ " символов");

return;

}

Account acc = adapter.getAccountByLogin(login);

if (acc.getId() != -1) {

registered = true;

}

if (acc.getPassword().equals(pass)) {

verifiedPass = true;

}

if (registered & verifiedPass) {

ArrayList<Children> list = adapter.getChildrensByAccount(acc);

Intent myIntent;

DataManager.setAccount(acc);

if (!adapter.isClosed())

adapter.close();

DataManager.getCurrentState().centerButtonClicked(getBaseContext(),

true, login);

} else if (!registered) {

alert = new AlertDialog.Builder(Login.this);

alert.setTitle("");

alert.setMessage("Введенной учетной записи не существует. Зарегистрировать её?");

alert.setPositiveButton("Да",

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

dialog.dismiss();

startRegistration();

}});

alert.setNegativeButton("Нет", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) { dialog.dismiss();

}});

alert.create().show();

} else if (!verifiedPass) {

password.setError("Неверный пароль");

}

}

private void startRegistration() {

adapter.close();

DataManager.getCurrentState().centerButtonClicked(getBaseContext(),

false, login);

}

private boolean isPasswordValid(String pass) {

return pass.length() >= \_passwordLength;

}

}

1. Код класса «Notes»

package forms;

import java.lang.reflect.Field;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Date;

import java.util.concurrent.TimeUnit;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataAdapter;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.ListViewNoteAdapter;

import com.example.babyprogressmap.MyAsyncTask;

import com.example.babyprogressmap.Note;

import com.example.babyprogressmap.R;

import com.example.babyprogressmap.R.id;

import com.example.babyprogressmap.R.layout;

import com.example.babyprogressmap.R.menu;

import android.app.ActionBar;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.DatePickerDialog;

import android.app.Dialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.PeriodicSync;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.os.Bundle;

import android.view.ContextMenu;

import android.view.ContextMenu.ContextMenuInfo;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewConfiguration;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemLongClickListener;

import android.widget.Button;

import android.widget.CalendarView;

import android.widget.DatePicker;

import android.widget.ImageButton;

import android.widget.ImageView;

import android.widget.ListView;

import android.widget.TextView;

public class Notes extends Activity {

ImageView imageView\_awatar;

TextView textView\_name;

TextView textView\_age;

TextView textView\_date;

Button button\_chooseDate;

ListView listView\_notes;

ImageButton imageButton\_add;

private int dlg\_year;

private int dlg\_month;

private int dlg\_day;

DataAdapter dataAdapter;

ListViewNoteAdapter listViewNoteAdapter;

static final int DATE\_DIALOG\_ID = 999;

@SuppressWarnings("deprecation")

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_notes\_form);

DataManager.getCurrentState().setActivity(this);

dataAdapter = new DataAdapter(this);

dataAdapter.open();

listViewNoteAdapter = new ListViewNoteAdapter(this);

imageView\_awatar = (ImageView) findViewById(R.id.imageView\_awatar);

textView\_name = (TextView) findViewById(R.id.textView\_name);

textView\_age = (TextView) findViewById(R.id.textView\_age);

textView\_date = (TextView) findViewById(R.id.textView\_date);

button\_chooseDate = (Button) findViewById(R.id.button\_chooseDate);

listView\_notes = (ListView) findViewById(R.id.listView\_notes);

imageButton\_add = (ImageButton) findViewById(R.id.imageButton\_add);

textView\_name.setText(DataManager.getChildren().getName());

imageButton\_add.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

DataManager.getCurrentState().middleButtonBarButtonClicked(getBaseContext());

}

});

Date dtChild = DataManager.getChildren().getBirthdate();

String age = getAge(dtChild);

textView\_age.setText(age);

Bitmap bmp = DataManager.getImageFromBytes(DataManager.getChildren()

.getAwatar());

imageView\_awatar.setImageBitmap(bmp);

button\_chooseDate.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

showDialog(DATE\_DIALOG\_ID);

}

});

SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy");

textView\_date.setText(format.format(DataManager.getChoosedDate()));

showNotes(DataManager.getChoosedDate());

listView\_notes.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> arg0, View arg1,int position, long id) {

DataManager.setNote(listViewNoteAdapter.getCollection()

.get(position));

}

});

listView\_notes.setLongClickable(true);

listView\_notes.setOnItemLongClickListener(new OnItemLongClickListener() {

@Override

public boolean onItemLongClick(AdapterView<?> arg0, View arg1, int position, long id) {

DataManager.setNote(listViewNoteAdapter.getCollection().get(position));

return false;

}

});

registerForContextMenu(listView\_notes);

ActionBar aBar = getActionBar();

aBar.setDisplayShowTitleEnabled(false);

aBar.setDisplayShowHomeEnabled(false);

aBar.show();

getOverflowMenu();

}

private void getOverflowMenu() {

try {

ViewConfiguration config = ViewConfiguration.get(this);

Field menuKeyField = ViewConfiguration.class

.getDeclaredField("sHasPermanentMenuKey");

if (menuKeyField != null) {

menuKeyField.setAccessible(true);

menuKeyField.setBoolean(config, false);

}

} catch (Exception e) {

e.printStackTrace();

}

}

@Override

public void onCreateContextMenu(ContextMenu menu, View v,

ContextMenuInfo menuInfo) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.context\_menu\_notes\_form\_, menu);

super.onCreateContextMenu(menu, v, menuInfo);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.action\_bar\_notes\_form, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.children\_profile:

DataManager.getCurrentState().childrenProfileClicked(

getBaseContext());

return true;

case R.id.change\_children\_profile:

DataManager.getCurrentState()

.changeProfileClicked(getBaseContext());

return true;

case R.id.logout:

DataManager.getCurrentState().logoutClicked(getBaseContext());

return true;

case R.id.notifications:

DataManager.getCurrentState()

.notificationsClicked(getBaseContext());

return true;

default:

return super.onOptionsItemSelected(item);

}

}

@SuppressWarnings("deprecation")

@Override

public boolean onContextItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.itemMaxsimize:

DataManager.getCurrentState().viewNoteClicked(getBaseContext());

break;

case R.id.itemEdit:

DataManager.getCurrentState().editNoteClicked(getBaseContext());

break;

case R.id.itemDelete:

AlertDialog alert = new AlertDialog.Builder(this).create();

alert.setTitle("");

alert.setMessage("Вы действительно хотите удалить эту заметку?");

alert.setButton("Да", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

dataAdapter.deleteNote(DataManager.getNote());

showNotes(DataManager.getNote().getPostdate());

}

});

alert.setButton2("Нет", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

// ...

}

});

alert.show();

break;

}

return super.onContextItemSelected(item);

}

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case DATE\_DIALOG\_ID:

final Calendar c = Calendar.getInstance();

int year = c.get(Calendar.YEAR);

int month = c.get(Calendar.MONTH);

int day = c.get(Calendar.DAY\_OF\_MONTH);

return new DatePickerDialog(this, datePickerListener, year, month,

day);

}

return null;

}

private void showNotes(Date date) {

ArrayList<Note> notes = dataAdapter.getNotesByDate(date, DataManager

.getChildren().getId());

listViewNoteAdapter.getCollection().clear();

for (Note note : notes) {

listViewNoteAdapter.addItem(note);

}

if (listViewNoteAdapter.getCollection().size() > 0) {

DataManager.setNote(listViewNoteAdapter.getCollection().get(0));

} else

DataManager.setNote(new Note());

listView\_notes.setAdapter(listViewNoteAdapter);

listViewNoteAdapter.notifyDataSetChanged();

}

public static long getDateDiff(Date date1, Date date2, TimeUnit timeUnit) {

long diffInMillies = date2.getTime() - date1.getTime();

return timeUnit.convert(diffInMillies, TimeUnit.MILLISECONDS);

}

public static int[] getDateDiffParts(Date date1, Date date2) {

long days = getDateDiff(date1, date2, TimeUnit.DAYS);

int year = (int) (days / 365);

int month = (int) ((days % 365) / 30);

int day = (int) ((days % 365) % 30);

return new int[] { year, month, day };

}

public String getAge(Date birthdate) {

long days = Math.abs(getDateDiff(new Date(), birthdate, TimeUnit.DAYS));

int year = (int) (days / 365);

int month = (int) ((days % 365) / 30);

int day = (int) ((days % 365) % 30);

String yearRes = "", monthRes = "", dayRes = "";

if (year != 0) {

if (("" + year).endsWith("1") & year != 11)

yearRes = year + " год";

else if (("" + year).endsWith("2") | ("" + year).endsWith("3")

| ("" + year).endsWith("4"))

yearRes = year + " года";

else

yearRes = year + " лет";

}

if (month != 0) {

if (("" + month).endsWith("1") & month != 11)

monthRes = month + " месяц";

else if (("" + month).endsWith("2") | ("" + month).endsWith("3")

| ("" + month).endsWith("4"))

monthRes = month + " месяца";

else

monthRes = month + " месяцев";

}

if (day != 0) {

if (("" + day).endsWith("1") & day != 11)

dayRes = day + " день";

else if (("" + day).endsWith("2") | ("" + day).endsWith("3")

| ("" + day).endsWith("4"))

dayRes = day + " дня";

else

dayRes = day + " дней";

}

return yearRes + " " + monthRes + " " + dayRes;

}

private DatePickerDialog.OnDateSetListener datePickerListener = new DatePickerDialog.OnDateSetListener() {

public void onDateSet(DatePicker view, int selectedYear,

int selectedMonth, int selectedDay) {

dlg\_year = selectedYear;

dlg\_month = selectedMonth;

dlg\_day = selectedDay;

int mString = dlg\_month + 1;

String resultDay = (dlg\_day + "").length() == 1 ? ("0" + dlg\_day)

: (dlg\_day + "");

String resultMonth = (mString + "").length() == 1 ? ("0" + mString)

: (mString + "");

textView\_date.setText(resultDay + "." + resultMonth + "."

+ dlg\_year);

Date dt = new Date(dlg\_year - 1900, dlg\_month, dlg\_day);

DataManager.setChoosedDate(dt);

showNotes(dt);

}

};

@Override

public void onBackPressed() {

}

}

1. Код класса «Notifications»

package forms;

import java.lang.reflect.Field;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Date;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.AlarmReceiver;

import com.example.babyprogressmap.DataAdapter;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.ListViewNoticeAdapter;

import com.example.babyprogressmap.Notice;

import com.example.babyprogressmap.R;

import com.example.babyprogressmap.R.id;

import com.example.babyprogressmap.R.layout;

import com.example.babyprogressmap.R.menu;

import android.app.ActionBar;

import android.app.Activity;

import android.app.AlarmManager;

import android.app.AlertDialog;

import android.app.PendingIntent;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.view.ContextMenu;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewConfiguration;

import android.view.ContextMenu.ContextMenuInfo;

import android.widget.AdapterView;

import android.widget.Button;

import android.widget.ImageButton;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.AdapterView.OnItemLongClickListener;

public class Notifications extends Activity {

ImageButton imageButton\_add;

ListView listView\_notifications;

DataAdapter dataAdapter;

ListViewNoticeAdapter listViewNoticeAdapter;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_notifications\_form);

DataManager.getCurrentState().setActivity(this);

imageButton\_add = (ImageButton) findViewById(R.id.imageButton\_add);

listView\_notifications = (ListView) findViewById(R.id.listView\_notifications);

dataAdapter = new DataAdapter(this);

if (dataAdapter.isClosed())

dataAdapter.open();

listViewNoticeAdapter = new ListViewNoticeAdapter(this);

showNotifications();

registerForContextMenu(listView\_notifications);

listView\_notifications

.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> arg0, View arg1,

int position, long id) {

DataManager.setNotice(listViewNoticeAdapter

.getCollection().get(position));

}

});

listView\_notifications.setLongClickable(true);

listView\_notifications

.setOnItemLongClickListener(new OnItemLongClickListener() {

@Override

public boolean onItemLongClick(AdapterView<?> arg0,

View arg1, int position, long id) {

DataManager.setNotice(listViewNoticeAdapter

.getCollection().get(position));

return false;

}

});

imageButton\_add.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

DataManager.getCurrentState().middleButtonBarButtonClicked(getBaseContext());

}

});

ActionBar aBar = getActionBar();

aBar.setDisplayShowTitleEnabled(false);

aBar.setDisplayShowHomeEnabled(false);

aBar.show();

getOverflowMenu();

}

private void showNotifications() {

ArrayList<Notice> notes = dataAdapter.getNotices();

listViewNoticeAdapter.getCollection().clear();

for (Notice notice : notes) {

listViewNoticeAdapter.addItem(notice);

}

listView\_notifications.setAdapter(listViewNoticeAdapter);

listViewNoticeAdapter.notifyDataSetChanged();

}

private void getOverflowMenu() {

try {

ViewConfiguration config = ViewConfiguration.get(this);

Field menuKeyField = ViewConfiguration.class

.getDeclaredField("sHasPermanentMenuKey");

if (menuKeyField != null) {

menuKeyField.setAccessible(true);

menuKeyField.setBoolean(config, false);

}

} catch (Exception e) {

e.printStackTrace();

}

}

private void deleteChoosedItem() {

int index = DataManager.getNotice().getId() + DataManager.MIN\_ALARM\_REQUEST\_ID;

Intent intent = new Intent(getBaseContext(),

AlarmReceiver.class);

PendingIntent sender = PendingIntent.getBroadcast(

getBaseContext(), index, intent,

PendingIntent.FLAG\_UPDATE\_CURRENT);

AlarmManager am = (AlarmManager) getSystemService(ALARM\_SERVICE);

am.cancel(sender);

dataAdapter.deleteNotice(DataManager.getNotice());

showNotifications();

}

@Override

public void onCreateContextMenu(ContextMenu menu, View v,

ContextMenuInfo menuInfo) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.context\_menu\_change\_notice, menu);

super.onCreateContextMenu(menu, v, menuInfo);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.action\_bar\_notifications\_form, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (!dataAdapter.isClosed())

dataAdapter.close();

switch (item.getItemId()) {

case R.id.notes:

DataManager.getCurrentState().notesClicked(getBaseContext());

return true;

case R.id.change\_children\_profile:

DataManager.getCurrentState()

.changeProfileClicked(getBaseContext());

return true;

case R.id.logout:

DataManager.getCurrentState().logoutClicked(getBaseContext());

return true;

case R.id.children\_profile:

DataManager.getCurrentState().viewChildrenProfileClicked(

getBaseContext());

return true;

default:

return super.onOptionsItemSelected(item);

}

}

@SuppressWarnings("deprecation")

public boolean onContextItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.edit:

DataManager.getCurrentState().editNoticeClicked(getBaseContext());

return true;

case R.id.delete:

AlertDialog alert = new AlertDialog.Builder(this).create();

alert.setTitle("");

alert.setMessage("Вы действительно хотите удалить выбранное уведомление?");

alert.setButton("Да", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

deleteChoosedItem();

}

});

alert.setButton2("Нет", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

// ...

}

});

alert.show();

return true;

}

return super.onContextItemSelected(item);

}

@Override

public void onBackPressed() {

}

}

1. Код класса «Registration»

package forms;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import com.example.babyprogressmap.Account;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.BabyProgressDataBaseHelper;

import com.example.babyprogressmap.DataAdapter;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.MyAsyncTask;

import com.example.babyprogressmap.R;

import com.example.babyprogressmap.R.id;

import com.example.babyprogressmap.R.layout;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.content.Intent;

import android.graphics.Color;

import android.os.Bundle;

import android.text.Editable;

import android.text.InputType;

import android.text.TextWatcher;

import android.view.View;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.DatePicker;

import android.widget.EditText;

import android.widget.LinearLayout;

public class Registration extends Activity {

private EditText edit\_name;

private EditText edit\_surname;

private EditText edit\_middlename;

private EditText edit\_login;

private EditText edit\_password;

private EditText edit\_passwordConfirm;

private DatePicker datepicker\_birthdate;

private Button button\_ok;

private Button button\_back;

BabyProgressDataBaseHelper helper;

int index = 0;

DataAdapter adapter;

private CheckBox checkBox\_showPassword;

private final int \_passwordLength = 6;

boolean passwordConfirmed = false;

boolean isUpdate;

@SuppressWarnings("deprecation")

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_registration\_hints);

DataManager.getCurrentState().setActivity(this);

edit\_name = (EditText) findViewById(R.id.edit\_name);

edit\_surname = (EditText) findViewById(R.id.edit\_surname);

edit\_middlename = (EditText) findViewById(R.id.edit\_middlename);

edit\_login = (EditText) findViewById(R.id.edit\_login);

edit\_password = (EditText) findViewById(R.id.edit\_password);

edit\_passwordConfirm = (EditText) findViewById(R.id.edit\_passwordConfirm);

datepicker\_birthdate = (DatePicker) findViewById(R.id.datePicker\_birthdate);

button\_ok = (Button) findViewById(R.id.button\_ok);

button\_back = (Button) findViewById(R.id.button\_back);

checkBox\_showPassword = (CheckBox) findViewById(R.id.checkBox\_showPassword);

adapter = new DataAdapter(this);

adapter.open();

Intent intent = getIntent();

isUpdate = intent.getBooleanExtra("isUpdate", false);

edit\_login.setText(intent.getStringExtra("login"));

if (isUpdate) {

edit\_login.setVisibility(1);

edit\_name.setText(DataManager.getAccount().getName());

edit\_surname.setText(DataManager.getAccount().getSurname());

edit\_middlename.setText(DataManager.getAccount().getMiddlename());

index = DataManager.getAccount().getId();

Date date = DataManager.getAccount().getBirthdate();

datepicker\_birthdate.init(date.getYear(), date.getMonth(),

date.getDate(), new DatePicker.OnDateChangedListener() {

@Override

public void onDateChanged(DatePicker view, int year,

int monthOfYear, int dayOfMonth) {

}

});

}

button\_ok.setOnClickListener(new View.OnClickListener() {

@SuppressWarnings("deprecation")

@Override

public void onClick(View v) {

try {

String message = "";

if (!isPasswordValid(edit\_password.getText().toString())) {

message = "Пароль должен быть не меньше "

+ \_passwordLength + " символов";

edit\_password.setError(message);

new MyAsyncTask(getApplicationContext())

.execute(message);

return;

}

if (!passwordConfirmed) {

message = "Пароли не свопадают";

edit\_passwordConfirm.setError(message);

new MyAsyncTask(getApplicationContext())

.execute(message);

return;

}

Account parent = new Account();

parent.setId(index);

parent.setName(edit\_name.getText().toString());

parent.setSurname(edit\_surname.getText().toString());

parent.setMiddlename(edit\_middlename.getText().toString());

parent.setLogin(edit\_login.getText().toString());

parent.setPassword(edit\_password.getText().toString());

// Date date = new Date();

//

// date.setDate(datepicker\_birthdate.getDayOfMonth());

// date.setMonth(datepicker\_birthdate.getMonth());

// date.setYear(datepicker\_birthdate.getYear());

Date date = new Date(datepicker\_birthdate.getYear() - 1900,

datepicker\_birthdate.getMonth(),

datepicker\_birthdate.getDayOfMonth());

parent.setBirthdate(date);

if (isUpdate)

adapter.updateParent(parent);

else {

adapter.insertParent(parent);

}

int id = adapter.getAccountId(parent.getLogin());

parent.setId(id);

DataManager.setAccount(parent);

adapter.close();

DataManager.getCurrentState().leftButtonBarButtonClicked(getBaseContext());

} catch (Exception e) {

e.printStackTrace();

}

}

});

button\_back.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

DataManager.getCurrentState().rightButtonBarButtonClicked(getBaseContext());

}

});

checkBox\_showPassword.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if (checkBox\_showPassword.isChecked()) {

edit\_password.setInputType(InputType.TYPE\_CLASS\_TEXT);

edit\_passwordConfirm

.setInputType(InputType.TYPE\_CLASS\_TEXT);

} else {

edit\_password.setInputType(InputType.TYPE\_CLASS\_TEXT

| InputType.TYPE\_TEXT\_VARIATION\_PASSWORD);

edit\_passwordConfirm.setInputType(InputType.TYPE\_CLASS\_TEXT

| InputType.TYPE\_TEXT\_VARIATION\_PASSWORD);

}

}

});

TextWatcher watcher = new TextWatcher() {

@Override

public void onTextChanged(CharSequence s, int start, int before,

int count) {

}

@Override

public void beforeTextChanged(CharSequence s, int start, int count,

int after) {

}

@Override

public void afterTextChanged(Editable s) {

if (edit\_password.getText().toString()

.equals(edit\_passwordConfirm.getText().toString())) {

edit\_passwordConfirm.setBackgroundColor(Color.GREEN);

passwordConfirmed = true;

} else {

edit\_passwordConfirm.setBackgroundColor(Color.RED);

passwordConfirmed = false;

}

edit\_passwordConfirm.refreshDrawableState();

}

};

edit\_passwordConfirm.addTextChangedListener(watcher);

edit\_password.addTextChangedListener(watcher);

}

@Override

public void onBackPressed() {

}

private boolean isPasswordValid(String pass) {

return pass.length() >= \_passwordLength;

}

}

1. Код класса «ViewChildrenProfile»

package forms;

import java.lang.reflect.Field;

import java.text.SimpleDateFormat;

import java.util.Date;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataAdapter;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.R;

import com.example.babyprogressmap.R.id;

import com.example.babyprogressmap.R.layout;

import com.example.babyprogressmap.R.menu;

import android.annotation.SuppressLint;

import android.app.ActionBar;

import android.app.Activity;

import android.content.Intent;

import android.graphics.Bitmap;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewConfiguration;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.TextView;

public class ViewChildrenProfile extends Activity {

TextView textView\_nameValue;

TextView textView\_surnameValue;

TextView textView\_middlenameValue;

TextView textView\_weightValue;

TextView textView\_growthValue;

TextView textView\_birthdateValue;

ImageView imageView\_Awatar;

Button button\_edit;

Button button\_back;

@SuppressLint("SimpleDateFormat")

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_children\_profile\_view);

DataManager.getCurrentState().setActivity(this);

textView\_nameValue = (TextView) findViewById(R.id.textView\_nameValue);

textView\_surnameValue = (TextView) findViewById(R.id.textView\_surnameValue);

textView\_middlenameValue = (TextView) findViewById(R.id.textView\_middlenameValue);

textView\_weightValue = (TextView) findViewById(R.id.textView\_weightValue);

textView\_growthValue = (TextView) findViewById(R.id.textView\_growthValue);

textView\_birthdateValue = (TextView) findViewById(R.id.textView\_birthdateValue);

imageView\_Awatar = (ImageView) findViewById(R.id.imageView\_Awatar);

button\_edit = (Button) findViewById(R.id.button\_edit);

button\_back = (Button) findViewById(R.id.button\_back);

textView\_nameValue.setText(DataManager.getChildren().getName());

textView\_surnameValue.setText(DataManager.getChildren().getSurname());

textView\_middlenameValue.setText(DataManager.getChildren()

.getMiddlename());

textView\_weightValue

.setText(DataManager.getChildren().getWeight() + "");

textView\_growthValue

.setText(DataManager.getChildren().getGrowth() + "");

Date date = DataManager.getChildren().getBirthdate();

SimpleDateFormat format = new SimpleDateFormat(

DataAdapter.DATE\_FORMAT\_RUS);

textView\_birthdateValue.setText(format.format(date));

Bitmap bmp = DataManager.getImageFromBytes(DataManager.getChildren()

.getAwatar());

imageView\_Awatar.setImageBitmap(bmp);

button\_edit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

DataManager.getCurrentState().leftButtonBarButtonClicked(

getBaseContext());

}

});

button\_back.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

DataManager.getCurrentState().rightButtonBarButtonClicked(

getBaseContext());

}

});

ActionBar aBar = getActionBar();

aBar.setDisplayShowTitleEnabled(false);

aBar.setDisplayShowHomeEnabled(false);

aBar.show();

getOverflowMenu();

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.action\_bar\_view\_children\_profile, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

Intent myIntent;

switch (item.getItemId()) {

case R.id.notes:

DataManager.getCurrentState().notesClicked(getBaseContext());

return true;

case R.id.change\_children\_profile:

DataManager.getCurrentState()

.changeProfileClicked(getBaseContext());

return true;

case R.id.notifications:

DataManager.getCurrentState().notificationsClicked(getBaseContext());

return true;

case R.id.logout:

DataManager.getCurrentState().logoutClicked(getBaseContext());

return true;

default:

return super.onOptionsItemSelected(item);

}

}

private void getOverflowMenu() {

try {

ViewConfiguration config = ViewConfiguration.get(this);

Field menuKeyField = ViewConfiguration.class

.getDeclaredField("sHasPermanentMenuKey");

if (menuKeyField != null) {

menuKeyField.setAccessible(true);

menuKeyField.setBoolean(config, false);

}

} catch (Exception e) {

e.printStackTrace();

}

}

@Override

public void onBackPressed() {

}

}

1. Код класса «ViewNote»

package forms;

import java.lang.reflect.Field;

import java.text.SimpleDateFormat;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataAdapter;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.Note;

import com.example.babyprogressmap.R;

import com.example.babyprogressmap.R.id;

import com.example.babyprogressmap.R.layout;

import com.example.babyprogressmap.R.menu;

import android.app.ActionBar;

import android.app.Activity;

import android.content.Intent;

import android.graphics.Bitmap;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewConfiguration;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.TextView;

public class ViewNote extends Activity {

ImageView imageView\_photo;

TextView textView\_tittle;

TextView textView\_description;

Button button\_edit;

Button button\_back;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_view\_note);

DataManager.getCurrentState().setActivity(this);

imageView\_photo = (ImageView) findViewById(R.id.imageView\_photo);

textView\_tittle = (TextView) findViewById(R.id.textView\_tittle);

textView\_description = (TextView) findViewById(R.id.textView\_description);

button\_edit = (Button) findViewById(R.id.button\_edit);

button\_back = (Button) findViewById(R.id.button\_back);

Note note = DataManager.getNote();

textView\_tittle.setText(note.getTitle());

textView\_description.setText(note.getDescription());

if (note.hasImage) {

Bitmap bmp = DataManager.getImageFromBytes(note.getPhoto());

imageView\_photo.setImageBitmap(bmp);

} else

imageView\_photo.setVisibility(View.GONE);

button\_edit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

DataManager.getCurrentState().leftButtonBarButtonClicked(

getBaseContext());

}

});

button\_back.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

DataManager.getCurrentState().rightButtonBarButtonClicked(

getBaseContext());

}

});

ActionBar aBar = getActionBar();

aBar.setDisplayShowTitleEnabled(false);

aBar.setDisplayShowHomeEnabled(false);

aBar.show();

getOverflowMenu();

}

private void getOverflowMenu() {

try {

ViewConfiguration config = ViewConfiguration.get(this);

Field menuKeyField = ViewConfiguration.class

.getDeclaredField("sHasPermanentMenuKey");

if (menuKeyField != null) {

menuKeyField.setAccessible(true);

menuKeyField.setBoolean(config, false);

}

} catch (Exception e) {

e.printStackTrace();

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.action\_bar\_view\_note, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.notes:

DataManager.getCurrentState().notesClicked(getBaseContext());

return true;

case R.id.change\_children\_profile:

DataManager.getCurrentState()

.changeProfileClicked(getBaseContext());

return true;

case R.id.notifications:

DataManager.getCurrentState()

.notificationsClicked(getBaseContext());

case R.id.logout:

DataManager.getCurrentState().logoutClicked(getBaseContext());

return true;

default:

return super.onOptionsItemSelected(item);

}

}

@Override

public void onBackPressed() {

}

}

#### ПРИЛОЖЕНИЕ В

Код классов из пакета «states»

1. Код класса «ChangeChildrenProfileState»

package states;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataManager;

import forms.ChangeChildrenProfile;

import forms.ChildrenProfile;

import forms.Login;

import forms.Notes;

import forms.Notifications;

import forms.ViewChildrenProfile;

import android.content.Context;

import android.content.Intent;

public class ChangeChildrenProfileState extends State {

/\*\*

\* @param 0 = context

\* \*/

@Override

public void middleButtonBarButtonClicked(Object... array) {

super.middleButtonBarButtonClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.ChangeChildrenProfile);

DataManager.setCurrentState(DataManager.getChildrenProfileState());

intent = new Intent(context, ChildrenProfile.class);

intent.putExtra(DataManager.Extra\_isUpdate, false);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = context

\* \*/

@Override

public void logoutClicked(Object... array) {

super.logoutClicked(array);

Context context = (Context) array[0];

DataManager.reset();

intent = new Intent(context, Login.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = context

\* \*/

@Override

public void notesClicked(Object... array) {

super.notesClicked(array);

DataManager.setPreviousActivity(ActivityEnum.ChangeChildrenProfile);

DataManager.setCurrentState(DataManager.getNotesState());

Context context = (Context) array[0];

intent = new Intent(context, Notes.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = context

\* \*/

@Override

public void viewChildrenProfileClicked(Object... array) {

super.viewChildrenProfileClicked(array);

DataManager.setPreviousActivity(ActivityEnum.ChangeChildrenProfile);

DataManager.setCurrentState(DataManager.getViewChildrenProfileState());

Context context = (Context) array[0];

intent = new Intent(context, ViewChildrenProfile.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = context

\* \*/

@Override

public void childrenProfileClicked(Object... array) {

super.childrenProfileClicked(array);

DataManager.setPreviousActivity(ActivityEnum.ChangeChildrenProfile);

DataManager.setCurrentState(DataManager.getChildrenProfileState());

Context context = (Context) array[0];

intent = new Intent(context, ChildrenProfile.class);

intent.putExtra(DataManager.Extra\_isUpdate, true);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = context

\* \*/

@Override

public void notificationsClicked(Object... array) {

super.notificationsClicked(array);

DataManager.setPreviousActivity(ActivityEnum.ChangeChildrenProfile);

DataManager.setCurrentState(DataManager.getNotificationsState());

Context context = (Context) array[0];

intent = new Intent(context, Notifications.class);

stateActivity.startActivity(intent);

}

}

1. Код класса «ChildrenProfileState»

package states;

import android.content.Context;

import android.content.Intent;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataManager;

import forms.ChangeChildrenProfile;

import forms.Notes;

import forms.Registration;

import forms.ViewChildrenProfile;

public class ChildrenProfileState extends State {

/\*\*

\* @param 0 = context

\* \*/

@Override

public void leftButtonBarButtonClicked(Object... array) {

super.leftButtonBarButtonClicked(array);

Context context = (Context) array[0];

switch (DataManager.getPreviousActivity()) {

case ChangeChildrenProfile:

intent = new Intent(context, ChangeChildrenProfile.class);

DataManager.setCurrentState(DataManager.getChangeChildrenProfileState());

break;

case Registration:

intent = new Intent(context, Notes.class);

DataManager.setCurrentState(DataManager.getNotesState());

break;

case ViewChildrenProfile:

intent = new Intent(context, ViewChildrenProfile.class);

DataManager.setCurrentState(DataManager.getViewChildrenProfileState());

break;

}

DataManager.setPreviousActivity(ActivityEnum.ChildrenProfile);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = context

\* \*/

@Override

public void rightButtonBarButtonClicked(Object... array) {

super.rightButtonBarButtonClicked(array);

Context context = (Context) array[0];

switch (DataManager.getPreviousActivity()) {

case Registration:

intent = new Intent(context, Registration.class);

DataManager.setCurrentState(DataManager.getRegistrationState());

break;

case ViewChildrenProfile:

intent = new Intent(context, ViewChildrenProfile.class);

DataManager.setCurrentState(DataManager.getViewChildrenProfileState());

break;

case ChangeChildrenProfile:

intent = new Intent(context, ChangeChildrenProfile.class);

DataManager.setCurrentState(DataManager.getChangeChildrenProfileState());

break;

}

DataManager.setPreviousActivity(ActivityEnum.ChildrenProfile);

stateActivity.startActivity(intent);

}

}

1. Код класса «EditNoteState»

package states;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataManager;

import forms.Notes;

import forms.ViewNote;

import android.content.Context;

import android.content.Intent;

public class EditNoteState extends State {

/\*\*

\* param 0 = (Context)context

\* \*/

@Override

public void leftButtonBarButtonClicked(Object... array) {

super.leftButtonBarButtonClicked(array);

Context context = (Context) array[0];

switch (DataManager.getPreviousActivity()) {

case ViewNote:

intent = new Intent(context, ViewNote.class);

DataManager.setCurrentState(DataManager.getViewNoteState());

break;

case Notes:

intent = new Intent(context, Notes.class);

DataManager.setCurrentState(DataManager.getNotesState());

break;

}

DataManager.setPreviousActivity(ActivityEnum.EditNote);

stateActivity.startActivity(intent);

}

/\*\*

\* param 0 = (Context)context

\* \*/

@Override

public void rightButtonBarButtonClicked(Object... array) {

super.rightButtonBarButtonClicked(array);

Context context = (Context) array[0];

switch (DataManager.getPreviousActivity()) {

case Notes:

intent = new Intent(context, Notes.class);

DataManager.setCurrentState(DataManager.getNotesState());

break;

case ViewNote:

intent = new Intent(context, ViewNote.class);

DataManager.setCurrentState(DataManager.getViewNoteState());

break;

}

DataManager.setPreviousActivity(ActivityEnum.EditNote);

stateActivity.startActivity(intent);

}

}

1. Код класса «EditNoticeState»

package states;

import android.content.Context;

import android.content.Intent;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataManager;

import forms.ChangeChildrenProfile;

import forms.Notifications;

public class EditNoticeState extends State {

@Override

public void leftButtonBarButtonClicked(Object... array) {

super.leftButtonBarButtonClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.EditNotice);

DataManager.setCurrentState(DataManager.getNotificationsState());

intent = new Intent(context, Notifications.class);

stateActivity.startActivity(intent);

}

@Override

public void rightButtonBarButtonClicked(Object... array) {

super.rightButtonBarButtonClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.EditNotice);

DataManager.setCurrentState(DataManager.getNotificationsState());

intent = new Intent(context, Notifications.class);

stateActivity.startActivity(intent);

}

}

1. Код класса «LoginState»

package states;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataManager;

import forms.ChangeChildrenProfile;

import forms.Registration;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

public class LoginState extends State {

/\*\*

\* @param 0 = context

\* @param 1 = registered

\* @param 2 = login

\* \*/

@Override

public void centerButtonClicked(Object... array) {

Context context = (Context) array[0];

boolean registered = (Boolean) array[1];

if (registered) // registered == true - незарегистрирован

{

intent = new Intent(context, ChangeChildrenProfile.class);

DataManager.setCurrentState(DataManager.getChangeChildrenProfileState());

} else {

intent = new Intent(context, Registration.class);

DataManager.setCurrentState(DataManager.getRegistrationState());

}

DataManager.setPreviousActivity(ActivityEnum.Login);

intent.putExtra(DataManager.Extra\_isUpdate, false);

intent.putExtra("login", array[2].toString());

stateActivity.startActivity(intent);

}

}

1. Код класса «NotesState»

package states;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataManager;

import com.example.babyprogressmap.MyAsyncTask;

import forms.ChangeChildrenProfile;

import forms.EditNote;

import forms.Login;

import forms.Notifications;

import forms.ViewChildrenProfile;

import forms.ViewNote;

import android.content.Context;

import android.content.Intent;

public class NotesState extends State {

/\*\*

\* @param 0 = (Context)context

\* \*/

@Override

public void middleButtonBarButtonClicked(Object... array) {

super.middleButtonBarButtonClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.Notes);

DataManager.setCurrentState(DataManager.getEditNoteState());

intent = new Intent(context, EditNote.class);

intent.putExtra(DataManager.Extra\_isUpdate, false);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\*/

@Override

public void logoutClicked(Object... array) {

super.logoutClicked(array);

Context context = (Context) array[0];

DataManager.reset();

intent = new Intent(context, Login.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\*/

@Override

public void changeProfileClicked(Object... array) {

super.changeProfileClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.Notes);

DataManager.setCurrentState(DataManager.getChangeChildrenProfileState());

intent = new Intent(context, ChangeChildrenProfile.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\*/

@Override

public void childrenProfileClicked(Object... array) {

super.viewChildrenProfileClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.Notes);

DataManager.setCurrentState(DataManager.getViewChildrenProfileState());

intent = new Intent(context, ViewChildrenProfile.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\*/

@Override

public void editNoteClicked(Object... array) {

super.editNoteClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.Notes);

DataManager.setCurrentState(DataManager.getEditNoteState());

intent = new Intent(context, EditNote.class);

intent.putExtra("isUpdate", true);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\*/

@Override

public void viewNoteClicked(Object... array) {

super.viewNoteClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.Notes);

DataManager.setCurrentState(DataManager.getViewNoteState());

intent = new Intent(context, ViewNote.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = context

\* \*/

@Override

public void notificationsClicked(Object... array) {

super.notificationsClicked(array);

DataManager.setPreviousActivity(ActivityEnum.Notes);

DataManager.setCurrentState(DataManager.getNotificationsState());

Context context = (Context) array[0];

intent = new Intent(context, Notifications.class);

stateActivity.startActivity(intent);

}

}

1. Код класса «NotificationsState»

package states;

import android.content.Context;

import android.content.Intent;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataManager;

import forms.ChangeChildrenProfile;

import forms.EditNotice;

import forms.Login;

import forms.Notes;

import forms.ViewChildrenProfile;

public class NotificationsState extends State{

@Override

public void logoutClicked(Object... array) {

super.logoutClicked(array);

Context context = (Context) array[0];

DataManager.reset();

intent = new Intent(context, Login.class);

stateActivity.startActivity(intent);

}

@Override

public void notesClicked(Object... array) {

super.notesClicked(array);

DataManager.setPreviousActivity(ActivityEnum.Notifications);

DataManager.setCurrentState(DataManager.getNotesState());

Context context = (Context) array[0];

intent = new Intent(context, Notes.class);

stateActivity.startActivity(intent);

}

@Override

public void changeProfileClicked(Object... array) {

super.changeProfileClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.Notifications);

DataManager.setCurrentState(DataManager.getChangeChildrenProfileState());

intent = new Intent(context, ChangeChildrenProfile.class);

stateActivity.startActivity(intent);

}

@Override

public void viewChildrenProfileClicked(Object... array) {

super.viewChildrenProfileClicked(array);

DataManager.setPreviousActivity(ActivityEnum.Notifications);

DataManager.setCurrentState(DataManager.getViewChildrenProfileState());

Context context = (Context) array[0];

intent = new Intent(context, ViewChildrenProfile.class);

stateActivity.startActivity(intent);

}

@Override

public void editNoticeClicked(Object... array) {

super.editNoticeClicked(array);

DataManager.setPreviousActivity(ActivityEnum.Notifications);

DataManager.setCurrentState(DataManager.getEditNoticeState());

Context context = (Context) array[0];

intent = new Intent(context, EditNotice.class);

intent.putExtra(DataManager.Extra\_isUpdate, true);

stateActivity.startActivity(intent);

}

@Override

public void middleButtonBarButtonClicked(Object... array) {

super.middleButtonBarButtonClicked(array);

DataManager.setPreviousActivity(ActivityEnum.Notifications);

DataManager.setCurrentState(DataManager.getEditNoticeState());

Context context = (Context) array[0];

intent = new Intent(context, EditNotice.class);

stateActivity.startActivity(intent);

}

}

1. Код класса «RegistrationState»

package states;

import android.content.Context;

import android.content.Intent;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataManager;

import forms.ChildrenProfile;

import forms.Login;

public class RegistrationState extends State {

/\*\*

\* @param 0 = (Context)context

\* \*/

@Override

public void leftButtonBarButtonClicked(Object... array) {

super.leftButtonBarButtonClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.Registration);

DataManager.setCurrentState(DataManager.getChildrenProfileState());

intent = new Intent(context, ChildrenProfile.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\* \*/

@Override

public void rightButtonBarButtonClicked(Object... array) {

super.rightButtonBarButtonClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.Registration);

DataManager.setCurrentState(DataManager.getLoginState());

intent = new Intent(context, Login.class);

stateActivity.startActivity(intent);

}

}

1. Код класса «State»

package states;

import com.example.babyprogressmap.DataManager;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import forms.Login;

//первым параметром всегда должен идти Context

public abstract class State {

protected Activity stateActivity;

protected Intent intent;

public void centerButtonClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void leftButtonBarButtonClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void middleButtonBarButtonClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void rightButtonBarButtonClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void logoutClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void notesClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void changeProfileClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void childrenProfileClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void editNoteClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void editNoticeClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void notificationsClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void viewChildrenProfileClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void viewNoteClicked(Object... array) {

Context context = (Context) array[0];

stateChanged(context);

};

public void setActivity(Activity \_this) {

stateActivity = \_this;

};

public void stateChanged(Context context) {

if (!DataManager.hasLogged()) {

DataManager.reset();

intent = new Intent(context, Login.class);

stateActivity.startActivity(intent);

}

}

}

1. Код класса «ViewChildrenProfileState»

package states;

import android.content.Context;

import android.content.Intent;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataManager;

import forms.ChangeChildrenProfile;

import forms.ChildrenProfile;

import forms.Login;

import forms.Notes;

import forms.Notifications;

public class ViewChildrenProfileState extends State {

/\*\*

\* @param 0 = (Context)context

\*/

@Override

public void leftButtonBarButtonClicked(Object... array) {

super.leftButtonBarButtonClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.ViewChildrenProfile);

DataManager.setCurrentState(DataManager.getChildrenProfileState());

intent = new Intent(context, ChildrenProfile.class);

intent.putExtra(DataManager.Extra\_isUpdate, true);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\*/

@Override

public void rightButtonBarButtonClicked(Object... array) {

super.rightButtonBarButtonClicked(array);

Context context = (Context) array[0];

switch (DataManager.getPreviousActivity()) {

case ChangeChildrenProfile:

intent = new Intent(context, ChangeChildrenProfile.class);

DataManager.setCurrentState(DataManager.getChangeChildrenProfileState());

break;

case Notes:

intent = new Intent(context, Notes.class);

DataManager.setCurrentState(DataManager.getNotesState());

break;

case ChildrenProfile:

intent = new Intent(context, ChildrenProfile.class);

intent.putExtra(DataManager.Extra\_isUpdate, true);

DataManager.setCurrentState(DataManager.getChildrenProfileState());

break;

case Notifications:

intent = new Intent(context, Notifications.class);

DataManager.setCurrentState(DataManager.getNotificationsState());

break;

}

DataManager.setPreviousActivity(ActivityEnum.ViewChildrenProfile);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\*/

@Override

public void logoutClicked(Object... array) {

super.logoutClicked(array);

Context context = (Context) array[0];

DataManager.reset();

intent = new Intent(context, Login.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\*/

@Override

public void notesClicked(Object... array) {

super.notesClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.ViewChildrenProfile);

DataManager.setCurrentState(DataManager.getNotesState());

intent = new Intent(context, Notes.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = context

\* \*/

@Override

public void changeProfileClicked(Object... array) {

super.changeProfileClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.ViewChildrenProfile);

DataManager.setCurrentState(DataManager.getChangeChildrenProfileState());

intent = new Intent(context, ChangeChildrenProfile.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = context

\* \*/

@Override

public void notificationsClicked(Object... array) {

super.notificationsClicked(array);

DataManager.setPreviousActivity(ActivityEnum.ViewChildrenProfile);

DataManager.setCurrentState(DataManager.getNotificationsState());

Context context = (Context) array[0];

intent = new Intent(context, Notifications.class);

stateActivity.startActivity(intent);

}

}

1. Код класса «ViewNoteState»

package states;

import android.content.Context;

import android.content.Intent;

import com.example.babyprogressmap.ActivityEnum;

import com.example.babyprogressmap.DataManager;

import forms.ChangeChildrenProfile;

import forms.EditNote;

import forms.Login;

import forms.Notes;

import forms.Notifications;

public class ViewNoteState extends State {

/\*\*

\* @param 0 = (Context)context

\* \*/

@Override

public void leftButtonBarButtonClicked(Object... array) {

super.leftButtonBarButtonClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.ViewNote);

DataManager.setCurrentState(DataManager.getEditNoteState());

intent = new Intent(context, EditNote.class);

intent.putExtra("isUpdate", true);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\* \*/

@Override

public void rightButtonBarButtonClicked(Object... array) {

super.rightButtonBarButtonClicked(array);

Context context = (Context) array[0];

switch (DataManager.getPreviousActivity()) {

case Notes:

intent = new Intent(context, Notes.class);

DataManager.setCurrentState(DataManager.getNotesState());

break;

case EditNote:

intent = new Intent(context, EditNote.class);

intent.putExtra(DataManager.Extra\_isUpdate, true);

DataManager.setCurrentState(DataManager.getEditNoteState());

break;

}

DataManager.setPreviousActivity(ActivityEnum.ViewNote);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\* \*/

@Override

public void logoutClicked(Object... array) {

super.logoutClicked(array);

Context context = (Context) array[0];

DataManager.reset();

intent = new Intent(context, Login.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\* \*/

@Override

public void notesClicked(Object... array) {

super.notesClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.ViewNote);

DataManager.setCurrentState(DataManager.getNotesState());

intent = new Intent(context, Notes.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = (Context)context

\* \*/

@Override

public void changeProfileClicked(Object... array) {

super.changeProfileClicked(array);

Context context = (Context) array[0];

DataManager.setPreviousActivity(ActivityEnum.ViewNote);

DataManager.setCurrentState(DataManager.getChangeChildrenProfileState());

intent = new Intent(context, ChangeChildrenProfile.class);

stateActivity.startActivity(intent);

}

/\*\*

\* @param 0 = context

\* \*/

@Override

public void notificationsClicked(Object... array) {

super.notificationsClicked(array);

DataManager.setPreviousActivity(ActivityEnum.ViewNote);

DataManager.setCurrentState(DataManager.getNotificationsState());

Context context = (Context) array[0];

intent = new Intent(context, Notifications.class);

stateActivity.startActivity(intent);

}

}

#### ПРИЛОЖЕНИЕ Г

Код файлов из папки «res/layout»

1. Код файла «activity\_change\_children\_profile.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"* >

<ListView

android:id=*"@+id/listView\_children"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_above=*"@+id/footerView"*

android:layout\_alignParentTop=*"true"* >

</ListView>

<RelativeLayout

android:id=*"@+id/footerView"*

style=*"@android:style/ButtonBar"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentBottom=*"true"*

android:layout\_centerHorizontal=*"true"* >

<ImageButton

android:id=*"@+id/imageButton\_add"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_centerInParent=*"true"*

android:layout\_centerVertical=*"true"*

android:background=*"@android:drawable/btn\_default\_small"*

android:src=*"@drawable/add\_clipart"* />

</RelativeLayout>

</RelativeLayout>

1. Код файла «activity\_children\_profile\_hints.xml»

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

tools:context=*".ChildrenProfile"* >

<ScrollView

android:id=*"@+id/activity\_schildren\_profile\_scroll"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:layout\_above=*"@+id/button\_bar"*

android:layout\_alignParentTop=*"true"* >

<LinearLayout

style=*"@style/textFieldsStyle"*

android:orientation=*"vertical"* >

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"vertical"* >

<TextView

android:id=*"@+id/textView\_title"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_horizontal"*

android:text=*"@string/children\_profile\_title"*

android:textAppearance=*"?android:attr/textAppearanceLarge"* />

<ImageView

android:id=*"@+id/imageView\_Awatar"*

android:layout\_width=*"150dp"*

android:layout\_height=*"150dp"*

android:layout\_gravity=*"center\_horizontal"*

android:src=*"@drawable/baby\_default"* />

<ImageButton

android:id=*"@+id/imageButton\_options"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_horizontal"*

android:src=*"@drawable/options"* />

</LinearLayout>

<EditText

android:id=*"@+id/edit\_name"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/name"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"text"*

android:maxLines=*"1"*

android:singleLine=*"true"* />

<EditText

android:id=*"@+id/edit\_surname"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/surname"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"text"*

android:maxLines=*"1"*

android:singleLine=*"true"* />

<EditText

android:id=*"@+id/edit\_middlename"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/middlename"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"text"*

android:maxLines=*"1"*

android:singleLine=*"true"* >

</EditText>

<EditText

android:id=*"@+id/edit\_weight"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/weight"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"numberDecimal"*

android:maxLines=*"1"*

android:singleLine=*"true"* />

<EditText

android:id=*"@+id/edit\_growth"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/growth"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"numberDecimal"*

android:maxLines=*"1"*

android:singleLine=*"true"* />

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical"*

android:orientation=*"vertical"* >

<TextView

android:id=*"@+id/textView\_birthdate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/birthdate"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<DatePicker

android:id=*"@+id/datePicker\_birthdate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_horizontal"* />

</LinearLayout>

</LinearLayout>

</ScrollView>

<LinearLayout

android:id=*"@+id/button\_bar"*

style=*"@style/ButtonBar"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentBottom=*"true"*

android:measureWithLargestChild=*"true"*

android:orientation=*"horizontal"* >

<LinearLayout

android:id=*"@+id/leftSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

<Button

android:id=*"@+id/button\_ok"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:background=*"@android:drawable/btn\_default"*

android:filterTouchesWhenObscured=*"true"*

android:maxLines=*"1"*

android:text=*"@string/Ok"*

android:textSize=*"8pt"*/>

<Button

android:id=*"@+id/button\_cancel"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:background=*"@android:drawable/btn\_default"*

android:maxLines=*"1"*

android:text=*"@string/return\_back"*

android:textSize=*"8pt"* />

<LinearLayout

android:id=*"@+id/rightSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

</LinearLayout>

</RelativeLayout>

1. Код файла «activity\_children\_profile\_view.xml»

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

tools:context=*".ViewChildrenProfile"* >

<ScrollView

android:id=*"@+id/activity\_schildren\_profile\_scroll"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:layout\_above=*"@+id/button\_bar"*

android:layout\_alignParentTop=*"true"* >

<LinearLayout

style=*"@style/textFieldsStyle"*

android:layout\_gravity=*"center\_vertical"*

android:orientation=*"vertical"* >

<ImageView

android:id=*"@+id/imageView\_Awatar"*

android:layout\_width=*"150dp"*

android:layout\_height=*"150dp"*

android:layout\_gravity=*"center\_horizontal"*

android:src=*"@drawable/baby\_default"* />

<LinearLayout

style=*"@style/textFieldsStyle"*

android:layout\_gravity=*"center\_vertical"*

android:orientation=*"horizontal"* >

<TextView

android:id=*"@+id/textView\_name"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginRight=*"5dp"*

android:text=*"@string/label\_name"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<TextView

android:id=*"@+id/textView\_nameValue"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/name"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

</LinearLayout>

<LinearLayout

style=*"@style/textFieldsStyle"*

android:layout\_gravity=*"center\_vertical"*

android:orientation=*"horizontal"* >

<TextView

android:id=*"@+id/textView\_surname"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginRight=*"5dp"*

android:text=*"@string/label\_surname"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<TextView

android:id=*"@+id/textView\_surnameValue"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/name"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

</LinearLayout>

<LinearLayout

style=*"@style/textFieldsStyle"*

android:orientation=*"horizontal"* >

<TextView

android:id=*"@+id/textView\_middlename"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginRight=*"5dp"*

android:text=*"@string/label\_middlename"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<TextView

android:id=*"@+id/textView\_middlenameValue"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/name"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

</LinearLayout>

<LinearLayout

style=*"@style/textFieldsStyle"*

android:layout\_gravity=*"center\_vertical"*

android:orientation=*"horizontal"* >

<TextView

android:id=*"@+id/textView\_weight"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginRight=*"5dp"*

android:text=*"@string/label\_weight"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<TextView

android:id=*"@+id/textView\_weightValue"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/name"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

</LinearLayout>

<LinearLayout

style=*"@style/textFieldsStyle"*

android:layout\_gravity=*"center\_vertical"*

android:orientation=*"horizontal"* >

<TextView

android:id=*"@+id/textView\_growth"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginRight=*"5dp"*

android:text=*"@string/label\_growth"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<TextView

android:id=*"@+id/textView\_growthValue"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/name"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

</LinearLayout>

<LinearLayout

style=*"@style/textFieldsStyle"*

android:layout\_gravity=*"center\_vertical"*

android:orientation=*"horizontal"* >

<TextView

android:id=*"@+id/textView\_birthdate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginRight=*"5dp"*

android:text=*"@string/label\_birthdate"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<TextView

android:id=*"@+id/textView\_birthdateValue"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/name"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

</LinearLayout>

</LinearLayout>

</ScrollView>

<LinearLayout

android:id=*"@+id/button\_bar"*

style=*"@style/ButtonBar"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentBottom=*"true"*

android:measureWithLargestChild=*"true"*

android:orientation=*"horizontal"* >

<!-- style="?android:attr/buttonBarStyle" -->

<LinearLayout

android:id=*"@+id/leftSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

<!-- android:layout\_marginTop="5dp" -->

<Button

android:id=*"@+id/button\_edit"*

style=*"@style/ButtonBarButton"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:filterTouchesWhenObscured=*"true"*

android:maxLines=*"1"*

android:text=*"@string/edit"*

android:textSize=*"8pt"*/>

<!-- style="?android:attr/buttonBarButtonStyle" -->

<Button

android:id=*"@+id/button\_back"*

style=*"@style/ButtonBarButton"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:background=*"@android:drawable/btn\_default\_small"*

android:maxLines=*"1"*

android:text=*"@string/return\_back"*

android:textSize=*"8pt"* />

<LinearLayout

android:id=*"@+id/rightSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

</LinearLayout>

</RelativeLayout>

1. Код файла «activity\_edit\_note.xml»

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

tools:context=*".ChildrenProfile"* >

<ScrollView

android:id=*"@+id/activity\_schildren\_profile\_scroll"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:layout\_alignParentTop=*"true"*

android:layout\_above=*"@+id/button\_bar"* >

<LinearLayout

xmlns:android=*"http://schemas.android.com/apk/res/android"*

style=*"@style/textFieldsStyle"*

android:orientation=*"vertical"* >

<ImageView

android:id=*"@+id/imageView\_photo"*

android:layout\_width=*"150dp"*

android:layout\_height=*"150dp"*

android:layout\_gravity=*"center\_horizontal"*

android:src=*"@drawable/baby\_default"* />

<ImageButton

android:id=*"@+id/imageButton\_options"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_horizontal"*

android:src=*"@drawable/options"* />

<EditText

android:id=*"@+id/editText\_title"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/title"*

android:maxLines=*"1"* >

<requestFocus />

</EditText>

<EditText

android:id=*"@+id/editText\_description"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/description"*

android:inputType=*"textMultiLine"* >

</EditText>

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical"*

android:orientation=*"vertical"* >

<TextView

android:id=*"@+id/textView\_postdate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/note\_date"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<DatePicker

android:id=*"@+id/datePicker\_postdate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_horizontal"* />

</LinearLayout>

</LinearLayout>

</ScrollView>

<LinearLayout

android:id=*"@+id/button\_bar"*

style=*"@style/ButtonBar"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentBottom=*"true"*

android:measureWithLargestChild=*"true"*

android:orientation=*"horizontal"* >

<LinearLayout

android:id=*"@+id/leftSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

<Button

android:id=*"@+id/button\_ok"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:background=*"@android:drawable/btn\_default"*

android:filterTouchesWhenObscured=*"true"*

android:maxLines=*"1"*

android:text=*"@string/Ok"*

android:textSize=*"8pt"*/>

<Button

android:id=*"@+id/button\_cancel"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:background=*"@android:drawable/btn\_default"*

android:maxLines=*"1"*

android:text=*"@string/return\_back"*

android:textSize=*"8pt"* />

<LinearLayout

android:id=*"@+id/rightSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

</LinearLayout>

</RelativeLayout>

1. Код файла «activity\_edit\_notice.xml»

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

tools:context=*".ChildrenProfile"* >

<ScrollView

android:id=*"@+id/activity\_schildren\_profile\_scroll"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:layout\_alignParentTop=*"true"*

android:layout\_above=*"@+id/button\_bar"* >

<LinearLayout

xmlns:android=*"http://schemas.android.com/apk/res/android"*

style=*"@style/textFieldsStyle"*

android:orientation=*"vertical"* >

<EditText

android:id=*"@+id/editText\_title"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/title"*

android:maxLines=*"1"* >

<requestFocus />

</EditText>

<EditText

android:id=*"@+id/editText\_description"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/description"*

android:inputType=*"textMultiLine"* >

</EditText>

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical"*

android:layout\_margin=*"10dp"*

android:orientation=*"vertical"* >

<TextView

android:id=*"@+id/textView\_posttime"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/time"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<TimePicker

android:id=*"@+id/timePicker\_posttime"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center"* />

</LinearLayout>

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical"*

android:layout\_margin=*"10dp"*

android:orientation=*"vertical"* >

<TextView

android:id=*"@+id/textView\_postdate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/date"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<DatePicker

android:id=*"@+id/datePicker\_postdate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center"* />

</LinearLayout>

</LinearLayout>

</ScrollView>

<LinearLayout

android:id=*"@+id/button\_bar"*

style=*"@style/ButtonBar"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentBottom=*"true"*

android:measureWithLargestChild=*"true"*

android:orientation=*"horizontal"* >

<LinearLayout

android:id=*"@+id/leftSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

<Button

android:id=*"@+id/button\_ok"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:background=*"@android:drawable/btn\_default"*

android:filterTouchesWhenObscured=*"true"*

android:maxLines=*"1"*

android:text=*"@string/Ok"*

android:textSize=*"8pt"* />

<Button

android:id=*"@+id/button\_back"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:background=*"@android:drawable/btn\_default"*

android:maxLines=*"1"*

android:text=*"@string/return\_back"*

android:textSize=*"8pt"* />

<LinearLayout

android:id=*"@+id/rightSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

</LinearLayout>

</RelativeLayout>

1. Код файла «activity\_login.xml»

<merge xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

tools:context=*".LoginActivity"* >

<!-- Login progress -->

<LinearLayout

android:id=*"@+id/login\_status"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center"*

android:gravity=*"center\_horizontal"*

android:orientation=*"vertical"*

android:visibility=*"gone"* >

<ProgressBar

style=*"?android:attr/progressBarStyleLarge"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginBottom=*"8dp"* />

<TextView

android:id=*"@+id/login\_status\_message"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginBottom=*"16dp"*

android:fontFamily=*"sans-serif-light"*

android:text=*"@string/login\_progress\_signing\_in"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

</LinearLayout>

<!-- Login form -->

<ScrollView

android:id=*"@+id/activity\_login\_scroll"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"* >

<LinearLayout

style=*"@style/textFieldsStyle"*

android:layout\_gravity=*"center\_vertical"*

android:orientation=*"vertical"* >

<EditText

android:id=*"@+id/email"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/login"*

android:inputType=*"textEmailAddress"*

android:maxLines=*"1"*

android:singleLine=*"true"* />

<EditText

android:id=*"@+id/password"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:hint=*"@string/password"*

android:imeActionId=*"@+id/login"*

android:imeActionLabel=*"@string/action\_sign\_in\_short"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"textPassword"*

android:maxLines=*"1"*

android:singleLine=*"true"* />

<Button

android:id=*"@+id/sign\_in\_button"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_horizontal"*

android:layout\_marginTop=*"16dp"*

android:paddingLeft=*"32dp"*

android:paddingRight=*"32dp"*

android:text=*"@string/sign\_or\_register"* />

</LinearLayout>

</ScrollView>

</merge>

1. Код файла «activity\_notes\_form.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"* >

<LinearLayout

android:id=*"@+id/linear\_header"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:background=*"@color/black"*

android:orientation=*"vertical"* >

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"horizontal"* >

<ImageView

android:id=*"@+id/imageView\_awatar"*

android:layout\_width=*"70dp"*

android:layout\_height=*"70dp"*

android:layout\_margin=*"5dp"*

android:scaleType=*"fitXY"*

android:src=*"@android:drawable/edit\_text"* />

<LinearLayout

xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"vertical"*

android:layout\_gravity=*"center\_vertical"* >

<TextView

android:id=*"@+id/textView\_name"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:maxLines=*"1"*

android:text=*"Name"*

android:textAppearance=*"?android:attr/textAppearanceLarge"*

android:textColor=*"@color/white"* />

<TextView

android:id=*"@+id/textView\_age"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:maxLines=*"1"*

android:text=*"3 года 2 месяца 1 день"*

android:textAppearance=*"?android:attr/textAppearanceMedium"*

android:textColor=*"@color/white"* />

</LinearLayout>

</LinearLayout>

<LinearLayout

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"horizontal"*

android:layout\_gravity=*"center\_horizontal"* >

<TextView

android:id=*"@+id/textView\_date"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:maxLines=*"1"*

android:text=*"01.01.2012"*

android:textAppearance=*"?android:attr/textAppearanceLarge"*

android:textColor=*"@color/white"* />

<Button

android:id=*"@+id/button\_chooseDate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/choose\_date"* />

</LinearLayout>

</LinearLayout>

<ListView

android:id=*"@+id/listView\_notes"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_above=*"@+id/relative\_footer"*

android:layout\_below=*"@+id/linear\_header"* >

</ListView>

<RelativeLayout

android:id=*"@+id/relative\_footer"*

style=*"@android:style/ButtonBar"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentBottom=*"true"*

android:orientation=*"vertical"* >

<ImageButton

android:id=*"@+id/imageButton\_add"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_centerHorizontal=*"true"*

android:layout\_centerVertical=*"true"*

android:background=*"@android:drawable/btn\_default\_small"*

android:src=*"@drawable/add\_clipart"* />

</RelativeLayout>

</RelativeLayout>

1. Код файла «activity\_notifications\_form.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"vertical"* >

<ListView

android:id=*"@+id/listView\_notifications"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_above=*"@+id/relativeLayout\_footer"*

android:layout\_below=*"@+id/relativeLayout\_header"* >

</ListView>

<RelativeLayout

android:id=*"@+id/relativeLayout\_footer"*

style=*"@android:style/ButtonBar"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentBottom=*"true"* >

<ImageButton

android:id=*"@+id/imageButton\_add"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_centerInParent=*"true"*

android:layout\_centerVertical=*"true"*

android:background=*"@android:drawable/btn\_default\_small"*

android:src=*"@drawable/add\_clipart"* />

</RelativeLayout>

</RelativeLayout>

1. Код файла «activity\_registration\_hints.xml»

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

tools:context=*".RegistrationActivity"* >

<ScrollView

android:id=*"@+id/activity\_schildren\_profile\_scroll"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:layout\_above=*"@+id/button\_bar"*

android:layout\_alignParentTop=*"true"* >

<LinearLayout

style=*"@style/textFieldsStyle"*

android:orientation=*"vertical"* >

<TextView

android:id=*"@+id/textView\_title"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_horizontal"*

android:text=*"@string/registration\_title"*

android:textAppearance=*"?android:attr/textAppearanceLarge"* />

<EditText

android:id=*"@+id/edit\_login"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"10dp"*

android:hint=*"@string/login"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"text"*

android:maxLines=*"1"*

android:singleLine=*"true"* >

<requestFocus />

</EditText>

<EditText

android:id=*"@+id/edit\_password"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"10dp"*

android:hint=*"@string/password"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"textPassword"*

android:maxLines=*"1"*

android:singleLine=*"true"* />

<EditText

android:id=*"@+id/edit\_passwordConfirm"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"10dp"*

android:hint=*"@string/confirm\_password"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"textPassword"*

android:maxLines=*"1"*

android:singleLine=*"true"* />

<CheckBox

android:id=*"@+id/checkBox\_showPassword"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"10dp"*

android:text=*"@string/show\_password"* />

<EditText

android:id=*"@+id/edit\_name"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"10dp"*

android:hint=*"@string/name"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"text"*

android:maxLines=*"1"*

android:singleLine=*"true"* >

</EditText>

<EditText

android:id=*"@+id/edit\_surname"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"10dp"*

android:hint=*"@string/surname"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"text"*

android:maxLines=*"1"*

android:singleLine=*"true"* />

<EditText

android:id=*"@+id/edit\_middlename"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"10dp"*

android:hint=*"@string/middlename"*

android:imeOptions=*"actionUnspecified"*

android:inputType=*"text"*

android:maxLines=*"1"*

android:singleLine=*"true"* >

</EditText>

<LinearLayout

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_horizontal"*

android:orientation=*"vertical"* >

<TextView

android:id=*"@+id/textView\_birthdate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/birthdate"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<DatePicker

android:id=*"@+id/datePicker\_birthdate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_horizontal"* />

</LinearLayout>

</LinearLayout>

</ScrollView>

<LinearLayout

android:id=*"@+id/button\_bar"*

style=*"@style/ButtonBar"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentBottom=*"true"*

android:measureWithLargestChild=*"true"*

android:orientation=*"horizontal"* >

<LinearLayout

android:id=*"@+id/leftSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

<Button

android:id=*"@+id/button\_ok"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:background=*"@android:drawable/btn\_default"*

android:filterTouchesWhenObscured=*"true"*

android:maxLines=*"1"*

android:text=*"@string/Ok"*

android:textSize=*"8pt"* />

<Button

android:id=*"@+id/button\_back"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:background=*"@android:drawable/btn\_default"*

android:maxLines=*"1"*

android:text=*"@string/return\_back"*

android:textSize=*"8pt"* />

<LinearLayout

android:id=*"@+id/rightSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

</LinearLayout>

</RelativeLayout>

1. Код файла «activity\_view\_note.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"* >

<ScrollView

android:id=*"@+id/activity\_schildren\_profile\_scroll"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:layout\_above=*"@+id/button\_bar"*

android:layout\_alignParentTop=*"true"* >

<LinearLayout

style=*"@style/textFieldsStyle"*

android:layout\_gravity=*"center\_vertical"*

android:orientation=*"vertical"* >

<ImageView

android:id=*"@+id/imageView\_photo"*

android:layout\_width=*"150dp"*

android:layout\_height=*"150dp"*

android:layout\_gravity=*"center\_horizontal"*

android:src=*"@drawable/baby\_default"* />

<TextView

android:id=*"@+id/textView\_tittle"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_horizontal"*

android:text=*"Large Text"*

android:textAppearance=*"?android:attr/textAppearanceLarge"* />

<TextView

android:id=*"@+id/textView\_description"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"10dp"*

android:text=*"Medium Text"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

</LinearLayout>

</ScrollView>

<LinearLayout

android:id=*"@+id/button\_bar"*

style=*"@style/ButtonBar"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentBottom=*"true"*

android:measureWithLargestChild=*"true"*

android:orientation=*"horizontal"* >

<!-- style="?android:attr/buttonBarStyle" -->

<LinearLayout

android:id=*"@+id/leftSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

<!-- android:layout\_marginTop="5dp" -->

<Button

android:id=*"@+id/button\_edit"*

style=*"@style/ButtonBarButton"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:filterTouchesWhenObscured=*"true"*

android:maxLines=*"1"*

android:text=*"@string/edit"* />

<!-- style="?android:attr/buttonBarButtonStyle" -->

<Button

android:id=*"@+id/button\_back"*

style=*"@style/ButtonBarButton"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical|center\_horizontal"*

android:layout\_weight=*"1"*

android:background=*"@android:drawable/btn\_default\_small"*

android:maxLines=*"1"*

android:text=*"@string/return\_back"* />

<LinearLayout

android:id=*"@+id/rightSpacer"*

android:layout\_width=*"0dip"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"0.25"*

android:orientation=*"horizontal"*

android:visibility=*"gone"* />

</LinearLayout>

</RelativeLayout>

1. Код файла «list\_item\_children.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"horizontal"* >

<ImageView

android:id=*"@+id/imageView\_awatar"*

android:layout\_width=*"50dp"*

android:layout\_height=*"50dp"*

android:layout\_margin=*"5dp"*

android:scaleType=*"fitXY"*

android:src=*"@android:drawable/edit\_text"* />

<TextView

android:id=*"@+id/textView\_name"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical"*

android:layout\_margin=*"5dp"*

android:text=*"some name"*

android:textAppearance=*"?android:attr/textAppearanceLarge"*

android:maxLines=*"1"* />

</LinearLayout>

1. Код файла «list\_item\_note\_without\_image.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*

android:layout\_margin=*"10dp"* >

<TextView

android:id=*"@+id/textView\_title"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Title"*

android:textAppearance=*"?android:attr/textAppearanceLarge"*

android:maxLines=*"1"* />

<TextView

android:id=*"@+id/textView\_description"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"description description description description description desc..."*

android:maxLines=*"2"* />

</LinearLayout>

1. Код файла «list\_item\_note.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"horizontal"* >

<ImageView

android:id=*"@+id/imageView"*

android:layout\_width=*"70dp"*

android:layout\_height=*"70dp"*

android:layout\_margin=*"5dp"*

android:scaleType=*"fitXY"*

android:src=*"@android:drawable/edit\_text"* />

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"vertical"* >

<TextView

android:id=*"@+id/textView\_title"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Title"*

android:textAppearance=*"?android:attr/textAppearanceLarge"*

android:maxLines=*"1"* />

<TextView

android:id=*"@+id/textView\_description"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"description description description description description desc..."*

android:maxLines=*"2"* />

</LinearLayout>

</LinearLayout>

1. Код файла «list\_item\_notice.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*>

<TextView

android:id=*"@+id/textView\_datetime"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"14:02 01.01.2013"*

android:textAppearance=*"?android:attr/textAppearanceLarge"*

android:layout\_gravity=*"center\_horizontal"* />

<TextView

android:id=*"@+id/textView\_title"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Title"*

android:textAppearance=*"?android:attr/textAppearanceMedium"*

android:layout\_gravity=*"center\_horizontal"* />

<TextView

android:id=*"@+id/textView\_description"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Descriptions"*

android:layout\_margin=*"8dp"* />

</LinearLayout>

#### ПРИЛОЖЕНИЕ Д

Код файлов из папки «res/menu»

1. Код файла «action\_bar\_change\_children\_profile.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item android:id=*"@+id/children\_profile"* android:title=*"@string/children\_profile"* android:showAsAction=*"never"*></item>

<item android:id=*"@+id/notes"* android:title=*"@string/notes"* android:showAsAction=*"never"*></item>

<item android:id=*"@+id/notifications"* android:title=*"@string/notifications"* android:showAsAction=*"never"*></item>

<item android:id=*"@+id/logout"* android:title=*"@string/logout"* android:showAsAction=*"never"*></item>

</menu>

1. Код файла «action\_bar\_notes\_form.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item android:id=*"@+id/children\_profile"* android:title=*"@string/children\_profile"* android:showAsAction=*"never"*></item>

<item android:id=*"@+id/change\_children\_profile"* android:title=*"@string/change\_children\_profile"* android:showAsAction=*"never"*></item>

<item android:id=*"@+id/notifications"* android:title=*"@string/notifications"* android:showAsAction=*"never"*></item>

<item android:id=*"@+id/logout"* android:title=*"@string/logout"* android:showAsAction=*"never"*></item>

</menu>

1. Код файла «action\_bar\_notifications\_form.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item

android:id=*"@+id/children\_profile"*

android:showAsAction=*"never"*

android:title=*"@string/children\_profile"*>

</item>

<item

android:id=*"@+id/change\_children\_profile"*

android:showAsAction=*"never"*

android:title=*"@string/change\_children\_profile"*>

</item>

<item

android:id=*"@+id/notes"*

android:showAsAction=*"never"*

android:title=*"@string/notes"*>

</item>

<item

android:id=*"@+id/logout"*

android:showAsAction=*"never"*

android:title=*"@string/logout"*>

</item>

</menu>

1. Код файла «action\_bar\_view\_children\_profile.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item android:id=*"@+id/change\_children\_profile"* android:title=*"@string/change\_children\_profile"* android:showAsAction=*"never"*></item>

<item android:id=*"@+id/notes"* android:title=*"@string/notes"* android:showAsAction=*"never"*></item>

<item android:id=*"@+id/notifications"* android:title=*"@string/notifications"* android:showAsAction=*"never"*></item>

<item android:id=*"@+id/logout"* android:title=*"@string/logout"* android:showAsAction=*"never"*></item>

</menu>

1. Код файла «action\_bar\_view\_note.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item

android:id=*"@+id/notes"*

android:showAsAction=*"never"*

android:title=*"@string/notes"*>

</item>

<item

android:id=*"@+id/change\_children\_profile"*

android:showAsAction=*"never"*

android:title=*"@string/change\_children\_profile"*>

</item>

<item

android:id=*"@+id/notifications"*

android:showAsAction=*"never"*

android:title=*"@string/notifications"*>

</item>

<item

android:id=*"@+id/logout"*

android:showAsAction=*"never"*

android:title=*"@string/logout"*>

</item>

</menu>

1. Код файла «context\_menu\_change\_children\_profile.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item android:id=*"@+id/edit"* android:title=*"@string/edit"*></item>

<item android:id=*"@+id/delete"* android:title=*"@string/delete"*></item>

</menu>

1. Код файла «context\_menu\_change\_notice.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item android:id=*"@+id/edit"* android:title=*"@string/edit"*></item>

<item android:id=*"@+id/delete"* android:title=*"@string/delete"*></item>

</menu>

1. Код файла «context\_menu\_children\_profile\_edit\_image.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item android:id=*"@+id/from\_camera"* android:title=*"@string/make\_photo"* android:icon=*"@drawable/camera\_clipart"*></item>

<item android:id=*"@+id/from\_gallery"* android:title=*"@string/pick\_from\_gallery"* android:icon=*"@drawable/gallery\_clipart"*></item>

</menu>

1. Код файла «context\_menu\_edit\_image\_options.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item android:id=*"@+id/from\_camera"* android:title=*"@string/make\_photo"* android:icon=*"@drawable/camera\_clipart"*></item>

<item android:id=*"@+id/from\_gallery"* android:title=*"@string/pick\_from\_gallery"* android:icon=*"@drawable/gallery\_clipart"*></item>

<item android:id=*"@+id/without\_image"* android:checkable=*"true"* android:title=*"@string/without\_image"* android:checked=*"false"* android:icon=*"@drawable/none\_clipart"*></item>

</menu>

1. Код файла «context\_menu\_notes\_form\_.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"* >

<item android:id=*"@+id/itemMaxsimize"* android:title=*"@string/maximize"*></item>

<item android:id=*"@+id/itemEdit"* android:title=*"@string/edit"*></item>

<item android:id=*"@+id/itemDelete"* android:title=*"@string/delete"*></item>

</menu>

#### ПРИЛОЖЕНИЕ Е

Код файлов из папки «res/values»

1. Код файла «strings.xml»

<resources>

<string name=*"wrong\_value"*>Некорректное значение</string>

<string name=*"app\_name"*>BabyProgressMap</string>

<string name=*"name"*>Имя</string>

<string name=*"surname"*>Фамилия</string>

<string name=*"middlename"*>Отчество</string>

<string name=*"birthdate"*>Дата рождения</string>

<string name=*"growth"*>Рост</string>

<string name=*"weight"*>Вес</string>

<string name=*"Apply"*>Применить</string>

<string name=*"Ok"*>Ок</string>

<string name=*"action\_sign\_in\_register"*><b>Sign in</b> or register</string>

<string name=*"action\_sign\_in\_short"*>Sign in</string>

<string name=*"menu\_forgot\_password"*>Восстановить пароль</string>

<string name=*"login\_progress\_signing\_in"*>Signing in…</string>

<string name=*"error\_invalid\_email"*>Неверный почтовый адрес</string>

<string name=*"error\_invalid\_password"*>This password is too short</string>

<string name=*"error\_incorrect\_password"*>This password is incorrect</string>

<string name=*"error\_field\_required"*>Обязательное поле</string>

<string name=*"sign\_or\_register"*>Войти или зарегистрироваться</string>

<string name=*"login"*>Имя учетной записи</string>

<string name=*"password"*>Пароль</string>

<string name=*"show\_password"*>Показать пароль</string>

<string name=*"confirm\_password"*>Подтверждение пароля</string>

<string name=*"choose"*>Выбрать</string>

<string name=*"edit"*>Изменить</string>

<string name=*"delete"*>Удалить</string>

<string name=*"add"*>Добавить</string>

<string name=*"title"*>Заголовок</string>

<string name=*"description"*>Описание</string>

<string name=*"make\_photo"*>Сделать фото</string>

<string name=*"pick\_from\_gallery"*>Взять из галереи</string>

<string name=*"without\_image"*>Без изображения</string>

<string name=*"children\_profile"*>Профиль ребенка</string>

<string name=*"change\_children\_profile"*>Выбор профиля</string>

<string name=*"exit"*>Закрыть</string>

<string name=*"logout"*>Выйти</string>

<string name=*"label\_name"*>Имя:</string>

<string name=*"label\_surname"*>Фамилия:</string>

<string name=*"label\_middlename"*>Отчество:</string>

<string name=*"label\_birthdate"*>Дата рождения:</string>

<string name=*"label\_growth"*>Рост:</string>

<string name=*"label\_weight"*>Вес:</string>

<string name=*"notes"*>Заметки</string>

<string name=*"note\_date"*>Дата заметки</string>

<string name=*"registration\_title"*>Учетная запись</string>

<string name=*"children\_profile\_title"*>Профиль ребенка</string>

<string name=*"choose\_date"*>Выбрать дату</string>

<string name=*"date"*>Дата</string>

<string name=*"time"*>Время</string>

<string name=*"cancel"*>Отмена</string>

<string name=*"return\_back"*>Назад</string>

<string name=*"notifications"*>Напоминалки</string>

<string name=*"maximize"*>На весь экран</string></resources>

1. Код файла «styles.xml»

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<resources xmlns:android=*"http://schemas.android.com/apk/res/android"*>

<style name=*"AppBaseTheme"* parent=*"android:Theme.Light"*></style>

<style name=*"AppTheme"* parent=*"AppBaseTheme"*></style>

<style name=*"textFieldsStyle"*>

<item name=*"android:layout\_width"*>match\_parent</item>

<item name=*"android:layout\_height"*>wrap\_content</item>

<item name=*"android:padding"*>10dp</item>

</style>

<style name=*"MyApp.Theme.Light.NoTitleBar"* parent=*"@android:style/Theme.Light.NoTitleBar"*>

<item name=*"android:textColorPrimaryInverse"*>@android:color/primary\_text\_light</item>

</style>

<style name=*"MyThemeButtonBar"* parent=*"@android:style/Theme"*></style>

<style name=*"ButtonBar"*>

<item name=*"android:paddingLeft"*>2dp</item>

<item name=*"android:paddingTop"*>5dp</item>

<item name=*"android:paddingRight"*>2dp</item>

<item name=*"android:paddingBottom"*>0dp</item>

<item name=*"android:background"*>@android:drawable/bottom\_bar</item>

</style>

<style name=*"ButtonBarButton"*>

<item name=*"android:paddingLeft"*>0dp</item>

<item name=*"android:background"*>@android:drawable/btn\_default\_small</item>

</style>

</resources>

Выпускная квалификационная работа выполнена мной самостоятельно. Использованные в работе материалы из опубликованной нормативно-справочной литературы, научных монографий и статей, а также других публикаций имеют ссылки на источники.

Отпечатано в 1 экземпляре,

библиография содержит 35 наименований,

количество страниц в основной части – 56,

количество страниц в приложениях – 99.

«\_\_» июня 2013 г.

Исполнитель,

студент гр. 9097 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Титов А.А.)

1. Единица измерения времени. Длительность определяется системой и обычно составляет 10 мс. [↑](#footnote-ref-1)
2. Форма представления информации, при которой содержимое (текст, изображение) двигается в вертикальном или горизонтальном направлении. Таким образом, скроллинг не изменяет содержимое, но передвигает «камеру». [↑](#footnote-ref-2)