

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий



## ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Тема: **Разработка кроссплатформенного  
мобильного приложения - клиента системы  
управления бизнес процессами**

Студент гр. 43501/3 Д.В. Круминьш



Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

Работа допущена к защите  
зав. кафедрой

\_\_\_\_\_ В.М. Ицыксон  
«\_\_\_\_» \_\_\_\_\_ 2017 г.

## ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Тема: **Разработка кроссплатформенного  
мобильного приложения - клиента системы  
управления бизнес процессами**

Направление: 230100 – Информатика и вычислительная техника

Выполнил студент гр. 43501/3 \_\_\_\_\_ Д.В. Круминьш  
Научный руководитель,  
к. т. н., доц. \_\_\_\_\_ И.В. Стручков



# РЕФЕРАТ

Отчет, 85 стр., 28 рис., 7 табл., 10 ист., 5 прил.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, БИЗНЕС СИСТЕМА, SAMUNDA,  
XAMARIN, C#, КРОССПЛАТФОРМЕННОСТЬ, МОБИЛЬНЫЕ  
ОПЕРАЦИОННЫЕ СИСТЕМЫ, ANDROID, IOS, BPM

В выпускной работе проводится разработка кроссплатформенного мобильного приложения - клиента для бизнес системы Camunda BPM. Обеспечивается запуск на следующих мобильных операционных системах:

- Android (версия 4.0.3 и старше);
- IOS (версия 6.1 и старше).

Функциональная часть приложения обеспечивает возможности модуля «Tasklist» системы Camunda BPM.

В работе проведен анализ рынка существующих систем автоматизации бизнес-процессов, их достоинств и недостатков. Был проведен анализ средств для разработки кроссплатформенного приложения, в результате чего был выбран фреймворк Xamarin. Средством разработки является объектно-ориентированный язык программирования C#. Взаимодействие между приложением и бизнес системой осуществляется посредством использования REST - интерфейса.



# СОДЕРЖАНИЕ

|   |           |
|---|-----------|
| <b>ВВЕДЕНИЕ . . . . .</b>   | <b>7</b>  |
| <b>1. Обзор существующих решений . . . . .</b>                            | <b>9</b>  |
| 1.1. Сравнение коммерческих и свободных систем . . . . .                  | 9         |
| 1.2. Мобильное приложение . . . . .                                       | 12        |
| 1.3. Camunda BPM . . . . .  | 13        |
| 1.3.1. Модуль Tasklist . . . . .  | 15        |
| 1.3.2. Мобильная версия . . . . .   | 15        |
| 1.4. Итоги . . . . .  | 17        |
| <b>2. Проектирование архитектуры и выбор средств разработки . . . . .</b> | <b>19</b> |
| 2.1. Постановка задачи . . . . .  | 19        |
| 2.2. REST - интерфейс . . . . .   | 19        |
| 2.3. Инструменты разработки кроссплатформенного приложения . . . . .      | 21        |
| 2.3.1. Native . . . . .   | 22        |
| 2.3.2. HTML5 . . . . .  | 22        |
| 2.3.3. PhoneGap . . . . .   | 23        |
| 2.3.4. Xamarin . . . . .  | 23        |
| 2.3.5. Сравнение PhoneGap и Xamarin . . . . .                             | 24        |
| 2.4. Итоги . . . . .  | 25        |
| <b>3. Проектирование и разработка системы . . . . .</b>                   | <b>27</b> |
| 3.1. Функциональность . . . . .   | 27        |
| 3.1.1. Подключение . . . . .  | 27        |
| 3.1.2. Работа с задачами . . . . .  | 28        |
| 3.1.3. Работа с процессами . . . . .                                      | 28        |
| 3.2. Кроссплатформенность составляющих . . . . .                          | 29        |
| 3.2.1. Работа с файлами . . . . .   | 29        |
| 3.2.2. Дизайн . . . . .   | 29        |
| 3.3. Реализация . . . . .   | 29        |
| 3.3.1. Структура классов и описание . . . . .                             | 30        |
| 3.3.2. POST и GET запросы . . . . .                                       | 32        |
| 3.3.3. Графический интерфейс . . . . .                                    | 35        |
| 3.4. Итоги . . . . .  | 37        |

|   |                             |    |
|---|-----------------------------|----|
| <b>4.</b>                               | <b>Тестирование системы</b> | 38 |
| 4.1.                                    | Backend тестирование        | 38 |
| 4.2.                                    | Frontend тестирование       | 39 |
| 4.2.1.                                  | Разбор теста                | 41 |
| 4.3.                                    | Итоги                       | 46 |
| <b>ЗАКЛЮЧЕНИЕ</b>                       |                             | 47 |
| <b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b> |                             | 48 |
| <b>ПРИЛОЖЕНИЕ А.</b>                    |                             | 50 |
| <b>ПРИЛОЖЕНИЕ Б.</b>                    |                             | 55 |
| <b>ПРИЛОЖЕНИЕ В.</b>                    |                             | 57 |
| <b>ПРИЛОЖЕНИЕ Г.</b>                    |                             | 59 |
| <b>ПРИЛОЖЕНИЕ Д.</b>                    |                             | 63 |

# **ВВЕДЕНИЕ**

Автоматизация бизнес-процессов в какой-либо компании - это внедрение программно-аппаратного комплекса на основе **BPM**, который обеспечивает качественное повышение уровня работы компании.

**BPM** (англ. Business Process Management, управление бизнес-процессами) — концепция процессного управления организацией, рассматривающая бизнес-процессы как особые ресурсы предприятия, непрерывно адаптируемые к постоянным изменениям, и полагающаяся на такие принципы, как понятность и видимость бизнес-процессов в организации.[1]

Основные причины внедрения системы автоматизации бизнес-процессов:

- Повышение скорости обработки и передачи информации между сотрудниками. Вся информация по какому-либо процессу будет доступна в одном месте;
- Каждое действие в бизнес-процессе, отмечено за определенным пользователем, впоследствии чего достигается прозрачность бизнес-процессов для всех участников;
- Минимизация затрат и ошибок;
- Общая видимость всех процессов в компании.

Главным отличием BPM-систем от других программ, умеющих выполнять построенные процессы, является возможность смоделировать и выполнить любой процесс предприятия, а не только специфический бизнес-процесс, связанный с продажами, документами или бухгалтерией.

Работа с большинством подобных систем происходит через web-интерфейс адаптированный для настольного ПК. Данное решение зачастую подходит для большинства задач, но в случае если необходима мобильность, то подобный подход несколько замедлит работу с системой. Наиболее оптимальным решением в данном случае является использование специально спроектированного мобильного

приложения для работы с бизнес-системой.

Целью данной работы является, создание мобильного приложения, обеспечивающего скорость и удобство сотрудников в системе ВРМ без привязки к рабочему месту в офисе, с сохранением полной функциональности настольной версии.

В рамках работы решаются следующие задачи:

1. Обзор существующих бизнес-систем, и их мобильных приложений;
2. Поиск наиболее оптимального средства для разработки кроссплатформенного мобильного приложения;
3. Разработка кроссплатформенного мобильного приложения - клиента системы управления бизнес процессами;
4. Тестирование разработанного приложения.

# 1. Обзор существующих решений

Практически любая система автоматизации бизнес-процессов имеет web-интерфейс, но все из них предлагают возможность работы на мобильном устройстве. Для обеспечения работы на мобильных устройствах существуют два подхода:

1. Оптимизация web-интерфейса для мобильных устройств;
2. Создания нативного клиента для мобильных операционных систем.

Наличие мобильного клиента свойственно коммерческим системам, в то время как свободно распространяемые системы, в лучшем случае имеют некоторую оптимизацию web-интерфейса для мобильных устройств.

## 1.1. Сравнение коммерческих и свободных систем

Коммерческие и свободные системы достаточно близки между собой по своему основному функционалу, но коммерческие системы помимо автоматизации бизнес-процессов, предлагают множество других возможностей, таких как:

- Почта;
- Чат и видеозвонки;
- Облачное хранение данных;
- и т.д.

Далее произведено сравнение бизнес систем, которые в первую очередь позиционируются как коммерческие. Сравнение было произведено по основным свойствам, имеющимся у каждой системы. Критерий стоимости определяют минимальную стоимость продукта, который включает:

1. Лицензию для установки на собственный сервер;
2. Возможность работы 25 пользователей.

|   | ELMA BPM   | Terrasoft BPM | Docsvision | Битрикс24  |
|---|------------|---------------|------------|------------|
| Не коммерческая версия  | ✓          | ✗             | ✗          | ✓          |
| Мобильный клиент  | ✓          | ✓             | ✓          | ✓          |
| Стоимость   | 165 000 р. | 893 750 р.    | 111 000 р. | 219 500 р. |
| Функциональные блоки систем BPM                                       |            |               |            |            |
| Кроссплатформенность мобильного клиента                               | ✓          | ✓             | ✗          | ✓          |
| Визуальный web-редактор бизнес-процессов                              | ✓          | ✓             | ✗          | ✓          |
| Поддержка стандарта BPMN 2.0  | ✓          | ✓             | ✓          | ✗          |
| Наличие интерфейса интеграции   | ✓          | ✓             | ✓*         | ✓          |
| Отчеты и графики по показателям эффективности работы бизнес-процессов | ✓          | ✓             | ✗          | ✓          |
| Простота переносимости между серверными платформами                   | ✗          | ✗             | ✗          | ✓          |

Таблица 1.1. Сравнение бизнес-систем

\*Имеется интеграция лишь с **1С:Предприятие** и **Microsoft Office**, которая предоставляется за дополнительную плату.

Системой с наименьшей стоимостью оказалась - **Docsvision**. Это произошло потому, что в отличии от прочих систем, в которых включен полный пакет возможностей в лицензию, в данной имеется возможность выбирать самостоятельно необходимые для работы модули.

Наличие не коммерческой версии хоть и является плюсом для её использования, но в тоже время такая версия имеет множество ограничений. Например подобная версия у **Битрикс24** не позволяет работать с бизнес-процессами, а у **ELMA BPM** не поддерживает существующий мобильный клиент.

Наиболее оптимальной коммерческой системой, с точки зрения возможностей и стоимости, является - **ELMA BPM**. Однако в ней имеется недостаток, в виде затрудненной переносимости системы между серверным платформами. В частности из-за использования платформы **.NET**, в следствии чего необходимо использовать платформу **MS Windows Server**.

Несмотря на стоимость коммерческих систем, недавно созданным компаниям с небольшим штатом сотрудников, подобные решения подходят больше, чем свободно распространяемые, по причине стоимости. С небольшим количеством пользователей стоимость лицензии будет соответственно меньше, в то время как затраты внедрение свободного ПО могут быть выше.

Также имеется возможность не покупать лицензию, а использовать облачный сервер, расположенный у компании производителя, что даст еще большую экономию средств.

С ростом компании, увеличивается и количество сотрудников в ней. Это приводит к существенному увеличению стоимости лицензии на бизнес-систему. В таком случае, для экономии средств, выбор остается за использованием свободно-распространяемых систем. К достоинствам подобных систем можно отнести:

1. Поддержка основных функций системы по автоматизации бизнес-процессов;
2. Размещение на собственных серверах;
3. Нет никаких ограничений на количество пользователей.

Таким образом, на начальных этапах работы компании, лучше использовать коммерческие решения. Смена коммерческого ПО на свободное, наиболее актуальна для крупных компаний с большим количеством сотрудников, в следствии чего существенно уменьшаются затраты на бизнес-систему.

## 1.2. Мобильное приложение

Согласно таблице 1.1, каждая система ориентированная на коммерческое использование, имеет соответствующий мобильный клиент, для платформ Android и IOS. Рассмотрим такой приложение на примере официального мобильного клиента **ELMA BPM** для **Android**.

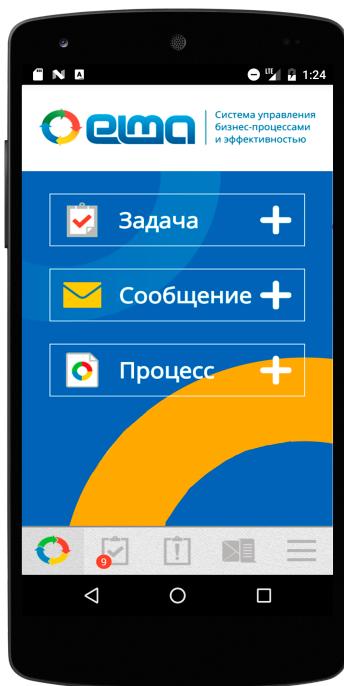


Рисунок 1.1. Главное меню приложения

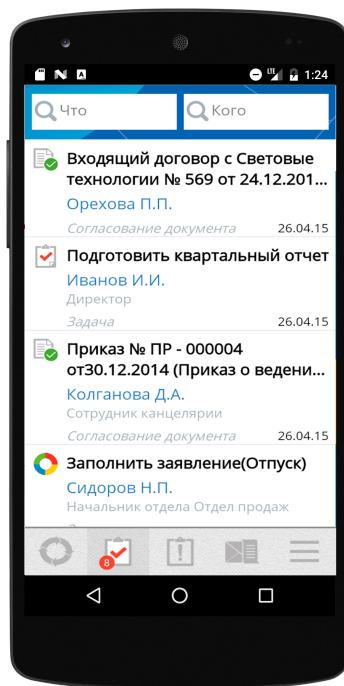


Рисунок 1.2. Список задач и процессов

На рисунке 1.1 представлено главное меню приложения, из которого имеется быстрый доступ к созданию задачи, процесса или отправке сообщения.

На рисунке 1.2 выведен список задач и процессов для текущего пользователя. Элементы списка, содержат краткую информацию о задаче. При раскрытии какого-либо элемента списка, предоставляется полная информация о задаче, а также возможности по взаимодействию:

- контроль задачи,
- комментирование,
- прикладывание файла.

К дополнительным возможностям приложения можно отнести работу в оффлайн режиме. Любые действия выполненные в данном режиме, будут автоматически переданы в систему при первом подключении.

Мобильный клиент **ELMA BPM** также доступен для **iPhone** и **iPad**.

### 1.3. Camunda BPM

Среди систем, распространяемых по свободной лицензии, наибольший потенциал представляет Camunda BPM, по следующим причинам:

1. Поддержка последних версий BPMN, CMMN, DMN;
2. Наличие инструментов для редактирования бизнес-процессов, мониторинга операций, управления персоналом;
3. Наличие объективных экспериментальных данных[2], о лучшей производительности Camunda в сравнении с другими BPMN системами;
4. Скорость развития, по сравнению с другими решениями;
5. Поддержка всех функциональных блоков из таблицы 1.1, за исключением мобильного клиента.

Camunda BPM является как открытой, так и коммерческой платформой, предоставляющей средства для автоматизации управления бизнес-процессами и организации рабочего процесса. В основе системы лежит движок моделирования бизнес-процессов, соответствующий концепции BPMN 2.0, который доступен в открытой версии.

Многие организации по всему миру используют Camunda для своих задач. Например "NASA Jet Propulsion Laboratory" использует Camunda с 2014 года.

За счет своей открытости, система имеет широкую поддержку от общества, за счет чего происходит наибольшее развитие проекта. Ближайшим конкурентом Camunda, с точки зрения открытости платформы, является Activity BPM. Однако данная система имеет меньшую популярность, в следствии чего она медленнее развивается.

При использовании интернет-ресурса <https://www.openhub.net>, была получена диаграмма (см. рис. 1.3), которая показывает количество написанных строк кода у обеих платформ. За период между

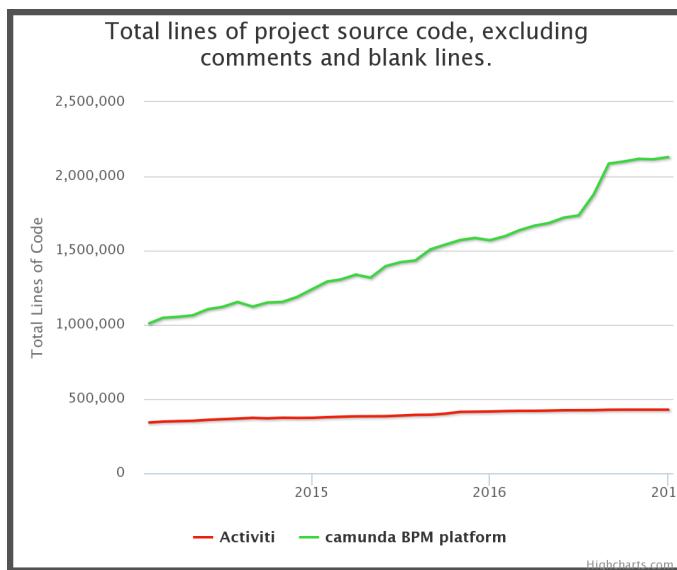


Рисунок 1.3. Сравнение Activity и Camunda по количеству написанных строк кода

февралем 2014 года и январем 2017 года, общее количество строк ко-

да у Camunda возросло с 1,009,616 до 2,129,369, а у Activity с 341,357 до 427,712 строк кода соответственно. Показатели подтверждают разницу в скорости развития между данными бизнес-системами.

Открытая версия Camunda состоит из следующих модулей:

- Admin - контроль над пользователями, группами, распределение привилегий доступа;
- Cockpit - мониторинг процессов и операций;
- Tasklist - работа с задачами и бизнес-процессами.

### **1.3.1. Модуль Tasklist**

Данный модуль необходим для обеспечения работы с задачами и бизнес-процессами, в частности, модуль позволяет:

- Создание произвольных задач или старт предварительно определенных бизнес-процессов;
- Получение списка текущих бизнес-процессов или задач, а также их состояния и возможностью просмотра более детальной информации;
- Разграничение задач по пользователям;
- Изменение статуса и комментирования бизнес-процессов или задач.

Вышеописанные действия доступны для пользователей данной системы, в соответствии с их правами доступа на те или иные операции. Интерфейс модуля, для настольного ПК, приведен на рисунке 1.4.

### **1.3.2. Мобильная версия**

Camunda BPM не имеет какого-либо нативного мобильного приложения, однако произведены некоторые оптимизации интерфейса для мобильных платформ. Для наглядности, данный модуль был открыт в браузере эмулятора мобильной операционной системы Android.

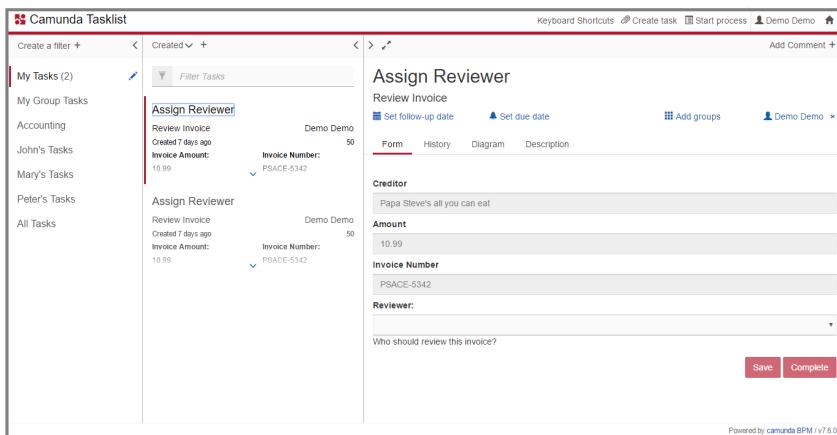


Рисунок 1.4. Интерфейс модуля Tasklist на настольном ПК

На рисунке 1.5 приведен общий вид интерфейса. Верхняя часть интерфейса с хаотично разбросанными элементами управления, наполовину перекрывает надпись "My Tasks(2)" что создает некоторые трудности для пользователя при взаимодействии с системой.

На рисунке 1.6 приведено всплывающее окно, при создании новой задачи. Напротив каждого поля присутствуют три символа точки, в настольной версии в данных местах расположено наименование поля для ввода. Подобное изменения текста произошло из-за невозможности полностью разместить весь текст на экране мобильного устройства.

Неполный показ текста также имеется при:

- Создании бизнес-процесса;
- Попытке применения фильтров;
- Просмотра подробной информации бизнес-процесса или задачи.

Таким образом, использование данного интерфейса на мобильном устройстве, сильно осложняет работу для пользователя, из-за невозможности показа необходимой информации.

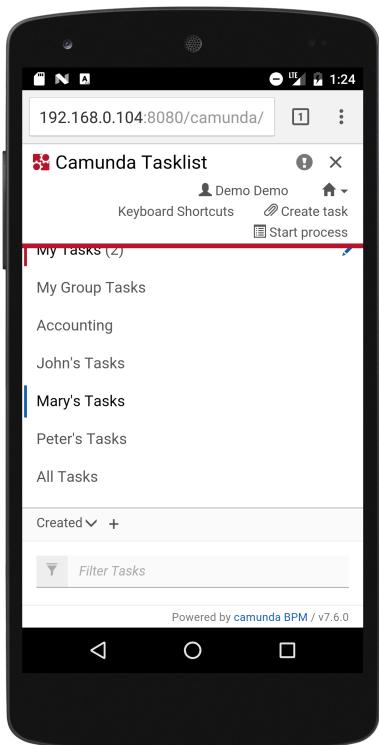


Рисунок 1.5. Общий вид

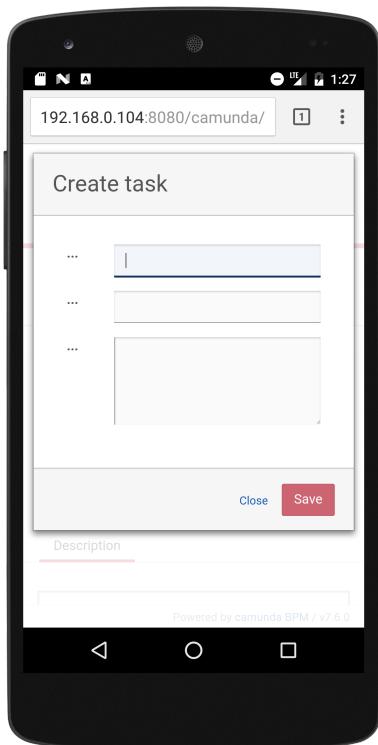


Рисунок 1.6. Создание задачи

## 1.4. Итоги

Были рассмотрены коммерческие системы, а также свободно-распространяемые на примере Camunda BPM. Сравнение, коммерческих систем, показало что:

- Каждая из них имеет мобильный клиент, чаще всего кроссплатформенный;
- Количество пользователей напрямую влияет на конечную стоимость лицензии;
- В большинстве из них имеются проблемы с переносимостью между серверными платформами.

Из-за зависимости стоимости лицензии от количества пользователей, крупным компаниям разумнее использовать свободное ПО.

Обзор мобильного клиента от **ELMA BPM** показал что, в нем присутствуют лишь наиболее важные функции, такие как:

- Создание задач и процессов;
- Просмотр списка задач, а также взаимодействия с ними;
- Отправка сообщений.

Обзор свободно-распространяемой бизнес-системы **Camunda BPM** показал что:

- Основные модули коммерческих систем есть и в Camunda;
- Имеется хорошая производительность системы;
- Из-за скорости развития имеется большой потенциал;
- Отсутствует какой-либо мобильный клиент. Текущая оптимизация web-интерфейса, для мобильных устройств, сильно осложняет работу для пользователя.

Таким образом, проектируемый кроссплатформенный мобильный клиент для **Camunda BPM** должен обеспечивать весь функционал модуля **Tasklist**, в частности:

- Взаимодействие и создание бизнес-процессов и задач;
- Разграничение предоставляемых данных, в соответствии с уровнем доступа пользователя;
- Возможность применять фильтры.

## 2. Проектирование архитектуры и выбор средств разработки

### 2.1. Постановка задачи

Необходимо провести разработку кроссплатформенного мобильного приложения - клиента для бизнес системы **Camunda BPM**, с поддержкой основных мобильных операционных систем (Android и IOS). Приложение должно обладать следующими функциями:

1. Старт бизнес-процесса или отдельной задачи;
2. Получение списка текущих бизнес-процессов или задач, а также их состояния и возможностью просмотра более детальной информации;
3. Действия с бизнес-процессами и задачами:
  - Закрепление за пользователем;
  - Изменение статуса;
  - Добавление новой переменной (текстового или числового формата);
  - Добавление нового комментария.

Взаимодействие между мобильным приложением и бизнес-системой, должно осуществляться посредством использования REST API.

### 2.2. REST - интерфейс

REST - это архитектурный стиль взаимодействия компонентов ПО в сети, посредством вызова стандартных HTTP методов.

В большинстве случаев вызываемые методы это **GET** или **POST** запросы, с некоторым набором параметров. В ответ на данные запросы, поступает код ответа и тело ответа.

Таким образом, одна транзакция данного API состоит из:

1. Метод запроса, например, GET
2. Путь запроса, например, /object/list

3. **Тело запроса**, например, данные в формате JSON
4. **Код ответа**, например, 200 OK
5. **Тело ответа**, например, данные в формате JSON

У Camunda также имеется поддержка данного API.

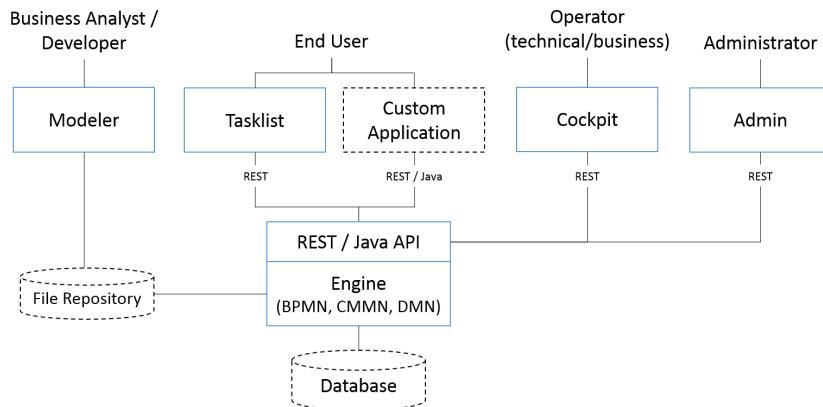


Рисунок 2.1. Архитектура Camunda

Из рисунка 2.1, видно что предоставлен доступ ко всем соответствующим интерфейсам системы, в том числе и к модулю Tasklist. Это позволяет проектировать сторонние приложения, способные взаимодействовать с данным модулем.

Для данного API, имеется полная документация [3], в которой для каждого используемого метода имеется:

- Описание метода;
- URL адрес метода;
- Описание возможных параметров запросов и ответов;
- Описание ответных кодов;
- Примеры запроса и ответа.

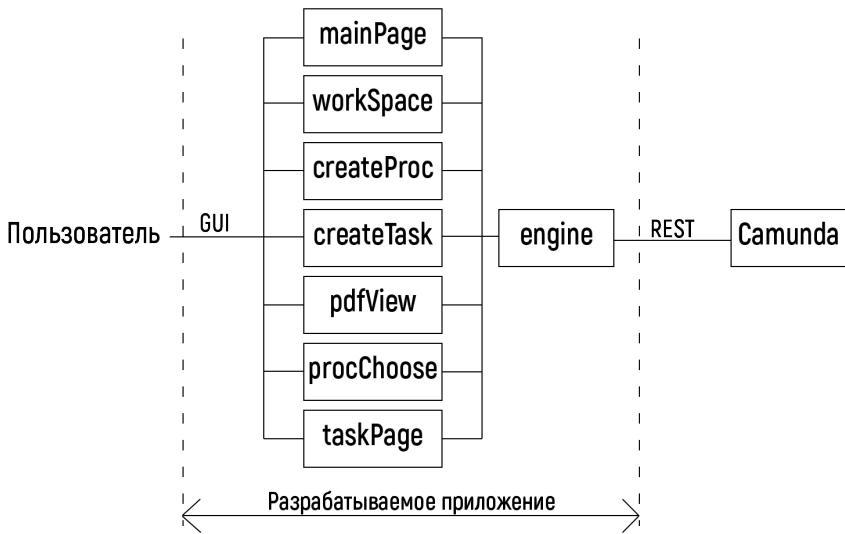


Рисунок 2.2. Архитектура разрабатываемого приложения

На рисунке 2.2 приведена архитектура разрабатываемого приложения. Пользователь, посредством GUI интерфейса, взаимодействует с соответствующими классами приложения, которые реализуют возможности модуля Tasklist. По результатам взаимодействия, вызываются соответствующие методы из класса `engine`, который уже посредством REST интерфейса взаимодействует с системой Camunda. Описание классов приведено в главе 3, таблице 3.1.

## 2.3. Инструменты разработки кроссплатформенного приложения

При создании мобильного ПО необходимо учитывать то, что в распоряжении пользователей имеются устройства на различных аппаратно-программных платформах. Таким образом поддержка большего количества платформ, даст больший охват пользователей.

На текущий момент существует множество инструментов для создания кроссплатформенного приложения, рассмотрим основные из них.

### **2.3.1. Native**

Создание нативного приложения для каждой из платформ в соответствующей среде разработки, имеет как достоинства, так и недостатки.

Достоинства:

1. Лучшая производительность и удобство использования конечно-го приложения;
2. Возможность использования всех платформозависимых возможностей;
3. Легкость распространения(публикация в магазине приложений соответствующей платформы) приложения;
4. Хорошая документация каждой из платформ.

К недостаткам можно отнести стоимость и время разработки, причинами этого является то, что для каждой из платформ используется свой язык программирования, в следствии чего возникнут сложности при переносе кода из одной системы в другую, особенно при работе с внешним видом приложения.

### **2.3.2. HTML5**

HTML5 является платформой для создания веб-приложений. Для разработки используется JavaScript, и так как он поддерживается всеми браузерами, то HTML5 будет работать на любом мобильном устройстве.

Достоинства:

1. Мгновенное обновление приложения;
2. Низкая стоимость и время разработки;

Недостатки:

1. Проблемы с использованием нативных функций устройств;
2. Трудности с распространением приложения;
3. Невозможно добиться нативного представления и адаптивности пользовательского интерфейса;

4. Проблемы с сохранением данных, при проблемах с подключением.

Является лучшим решением при недостатки бюджета или необходимости сделать приложение в короткие сроки.

### **2.3.3. PhoneGap**

PhoneGap - это OpenSource платформа, которая использует возможности HTML5, но с добавлением поддержки нативных возможностей платформы.

Достоинства:

1. Расширено API браузера по сравнению с HTML5, что дает доступ к нативным возможностям;
2. Легкость распространения приложения.

Недостатки:

1. Увеличение затрат на разработку, по сравнению с HTML5;
2. По-прежнему невозможно добиться нативного представления и адаптивности пользовательского интерфейса, так как визуализация происходит с помощью встроенного браузера.

### **2.3.4. Xamarin**

Xamarin — это фреймворк для кроссплатформенной разработки мобильных приложений с использованием языка C#.

Достоинства:

1. Все нативные возможности;
2. Нативное представление и адаптивность пользовательского интерфейса;
3. Приложения на разных системах, будут выглядеть очень похоже;
4. Легкость распространения приложения.

Недостатки:

1. Увеличенный размер конечного приложения, из-за необходимости вложения в приложение библиотек для его работы;
2. Если разработка происходит на ОС Windows, то для мобильной системы IOS необходима ОС MAC.

Xamarin состоит из множества частей, так при проектировке интерфейса имеется два подхода:

1. Использование **Xamarin.Forms** для создание единого интерфейса для все платформ;
2. Создание одного интерфейса для одной платформы, используя соответствующие инструменты субплатформы:
  - **Xamarin.Android**
  - **Xamarin.IOS**
  - **Windows Phone**

Использование Xamarin.Forms позволит дополнительно сократить время разработки, при условии использования простого интерфейса. В случае если необходимо использовать какие-либо платформозависимые графические элементы, то необходимо проектировать интерфейс для каждой из платформ отдельно.

Подобная возможность касается и основного кода приложения, при необходимости можно использовать платформозависимые функции системы.

Конечное приложение при использовании Xamarin практически не уступают нативным приложениям, которые были сразу написаны для этой платформы, за исключением увеличенного размера приложения, при этом существенно выигрывая в стоимости и времени разработки.

### 2.3.5. Сравнение PhoneGap и Xamarin

По итогам вышеизложенных обзоров, можно выделить два решения которые позволяют сократить время и стоимость разработки, сохранив функциональность приложения. Дополнительно сравним их между собой.

|                             | Xamarin  | PhoneGap  |
|-----------------------------|--|---|
| Язык                        | C#   | HTML5, CSS, JavaScript  |
| UI                          | Нативный   | Ограниченный  |
| Доступ к API устройства     | Все нативные возможности платформы уже доступны. Нет необходимости использовать что-то дополнительное. | Ограниченнная поддержка.<br>Необходимость написание нативно зависимых частей на нативном языке платформы. |
| Поддерживаемые платформы    | Android, IOS, Windows Phone, Windows 8/RT, Tizen   | Android, IOS, Windows Phone, Blackberry, WebOS, Symbian, Bada, Ubuntu, Firefox OS                         |
| Нативная производительность | Стабильная работа на всех платформах   | Более медленная и худшая производительность   |
| Время разработки            | Разумное   | Разумное  |

Таблица 2.1. Сравнение кроссплатформенных решений

Из таблицы 2.1 следует что Xamarin имеет лучшую производительность, отзывчивость интерфейса, а также полную функциональность при работе с API устройства.

Наибольшим недостатком PhoneGap является то, что конечное приложение запускается внутри браузера(внутри элемента WebView), что и влечет за собой низкую производительность.

У Xamarin подобная проблема не наблюдается.

## 2.4. Итоги

Были рассмотрены различные решения для создания кроссплатформенных приложений. Проектирование нативных приложений, для каждой из платформ, даст наилучший результат. Однако в данном случае стоимость и время такой разработки будет максимальными.

Компромиссом являются решения, позволяющие проектировать приложения сразу для нескольких платформ.

Наилучшим решением, с точки зрения "цена/качество", является Xamarin. Его использование позволяет существенно сократить количество требуемого для написания кода, в тоже время сохраняя весь потенциал возможностей конечной платформы.

Для взаимодействия разрабатываемого приложения и системы Camunda, был выбран REST - интерфейс.

### 3. Проектирование и разработка системы

Разработка происходила на языке программирования C#, с использованием кроссплатформенного фреймворка **Xamarin**. Для реализации графического интерфейса был выбран **Xamarin.Forms**. Конечные мобильные платформы:

- **Android**
- **iOS.**

#### 3.1. Функциональность

Возможности модуля **Tasklist** web-версии системы, были перенесены в кроссплатформенное мобильное приложение, в котором имеется:

1. Экран для ввода аутентификационной информации пользователя и данных для подключения к серверу;
2. Главный экран, который содержит:
  - Данные пользователя(Имя, Фамилия, E-mail);
  - Кнопки создания задачи и процесса;
  - Кнопку выхода.
3. Экран со списком всех задач и процессов отсортированных по дате(новые выше);
4. Экран со списком задач и процессов, закрепленных за текущим пользователем.

##### 3.1.1. Подключение

Для подключения к бизнес системе, пользователю необходимо ввести:

1. Адрес сервера;
2. Порт сервера для подключения;

3. Имя пользователя;
4. Пароль пользователя.

В случае успешного подключения, пользователь попадает в основное меню приложения для дальнейших действий.

В случае неудачного подключения, пользователю выводится соответствующие сообщение об ошибке. Такими ошибками могут быть:

1. Неверный логин или пароль;
2. Недоступность сервера;
3. Ошибка тайм-аута;
4. Неизвестная ошибка.

### **3.1.2. Работа с задачами**

При создании задачи, пользователь необходимо ввести:

1. Название задачи;
2. Описание задачи.

Далее имеется возможность производить следующие действия:

1. Закреплять или откреплять эту задачу за текущим пользователем;
2. Просматривать и добавлять комментарии к задаче (комментарии отсортированы по дате их добавления);
3. Перевод задачи в состояние завершенной;
4. Добавление собственных переменных.

### **3.1.3. Работа с процессами**

Для того чтобы создать новый процесс, пользователю предоставляется выбор имеющихся в системе процессов. После выбора процесса, на экране устройства выводятся\* поля для заполнения, которые уже предварительно заданы для данного процесса.

\*Если начальная форма этого процесса типа **Generated Task Forms**, иначе выводится соответствующие сообщение о невозможности создать процесс.

Все возможности при работе с задачами так-же применимы для работы с процессами.

Помимо обычных текстовых полей, к процессам могут быть прикреплены файлы с расширением .pdf. Для того чтобы отобразить подобный файл, была использована свободно-распространяемая библиотека **pdf.js**. Которая позволяет отобразить подобный файл прямо в приложении.

## 3.2. Кроссплатформенность составляющих

Большая часть кода написанного приложения является кроссплатформенной, однако часть дизайна и работа с файлами, потребовали использования платформозависимого кода.

### 3.2.1. Работа с файлами

Так как процесс может содержать файлы с расширением **.pdf**, то перед тем как открыть файл, его необходимо загрузить на устройство. В качестве паки для загрузки файлов, выбрана персональная папка приложения. То-есть например на устройстве android, чтобы извне получить доступ к этому файлу, потребуются права администратора - root.

Еще одной частью при работе с файлами, являлось открытия скачанного файла с помощью библиотеки **pdf.js**. В данном случае различались методы загрузки файла .pdf в элемент **CustomWebView**.

### 3.2.2. Дизайн

Не все графические элементы отображаются как задумано на конечных мобильных операционных системах. Так например для **iOS** требуется дополнительный отступ сверху, для того чтобы не залезать на **statusBar** системы. Поэтому в зависимости от системы, в графические элементы вносятся, соответствующие изменения.

## 3.3. Реализация

Большая часть приложения состоит в корректной обработке POST и GET запросов.

### 3.3.1. Структура классов и описание

Приложение состоит из 22 классов и 1 интерфейса (ISave(для работы с файловой системой платформы)). Далее приведена UML диаграмма и таблица с этими классами и их описанием.

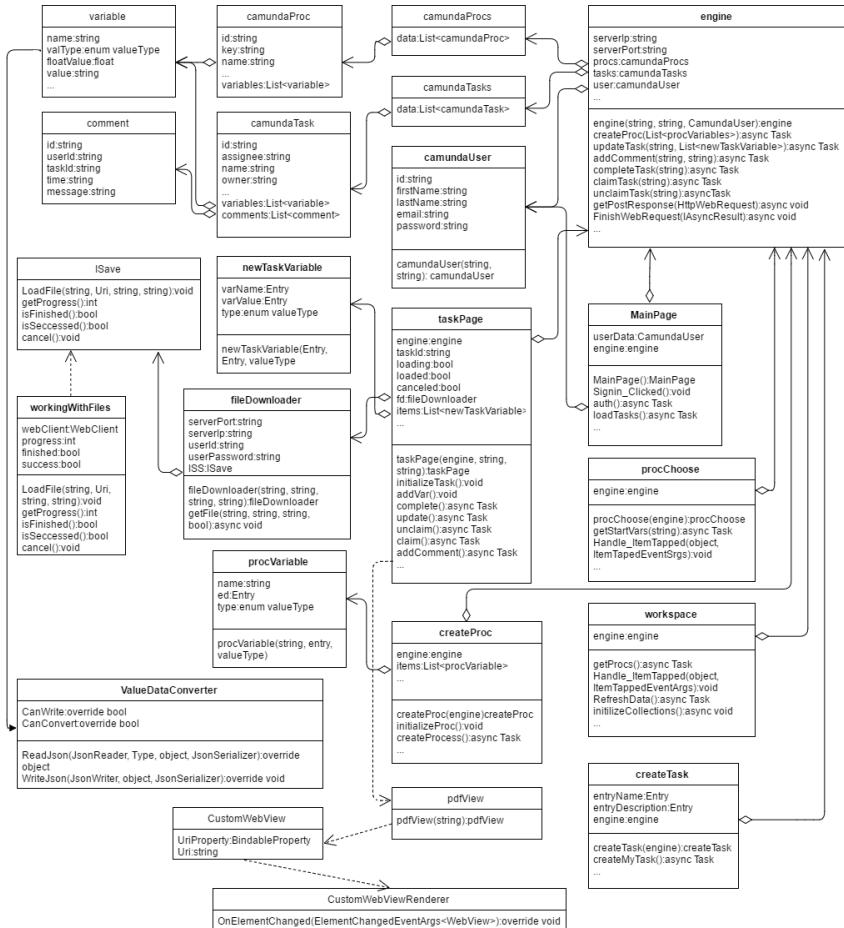


Рисунок 3.1. UML диаграмма классов

| Название класса    | Описание   |
|--------------------|--|
| camundaProc        | Описание свойств возможных для запуска процессов   |
| camundaProcs       | Список объектов класса типа camundaProc  |
| camundaTask        | Описание свойств уже запущенного процесса или задачи   |
| camundaTasks       | Список объектов класса типа camundaTask  |
| comment            | Описание свойств комментария   |
| camundaUser        | Данные пользователя  |
| createProc         | GUI - класс для создания нового процесса   |
| createTask         | GUI - класс для создания новой задачи  |
| CustomWebView      | Элемент для отображения файла pdf  |
| engine             | Все взаимодействия (GET и POST запросы) между приложением и сервером происходят в данном классе      |
| MainPage           | GUI - класс для ввода идентификационной информации для подключения к серверу                         |
| pdfView            | GUI - класс для отображения файлов формата pdf   |
| procChoose         | GUI - класс, который предоставляет пользователю выбор нового процесса                                |
| taskPage           | GUI - класс для отображения выбранной задачи или процесса  |
| newTaskVariable    | Хранение новых переменных добавленных пользователем  |
| procVariable       | Хранение значений начальных элементов при создании процесса  |
| ValueDataConverter | Переопределение некоторых методов класса <b>JsonConverter</b> для корректного анализа входной строки |

|                           |   |
|---------------------------|---|
| variable                  | Хранение уже имеющихся переменных задачи или процесса   |
| workspace                 | GUI - класс, который позволяет создавать новые задачи, просматривать имеющиеся задачи                                     |
| fileDownloader            | Общий класс для организации загрузки pdf файла  |
| Платформозависимые классы |   |
| workingWithFiles          | Работа с файловой системой, для загрузки файлов. Также обеспечивает возможность просмотра прогресса загрузки и её отмены. |
| CustomWebViewRenderer     | Загрузка файла pdf в элемент класса CustomWebView   |

Таблица 3.1. Классы и их описания

### 3.3.2. POST и GET запросы

Рассмотрим формирование **GET** запроса, для аутентификации пользователя.

Листинг 3.1. Метод **authenticate** класса **engine**.

```

1 public void authenticate(){
2     req = null;
3     req = (HttpWebRequest)WebRequest.Create(@"http://" + serverIp +
4         ":" + serverPort + "/engine-rest/user/" + User.Id +
5         "/profile");
6     req.Method = "GET";
7     req.Headers["Authorization"] = "Basic " + Convert.ToBase64String(
8         Encoding.UTF8.GetBytes(User.Id + ":" + User.Password));
9
10    // Adding type of response what are we waiting for
11    processType pt = processType.userInfo;
12    // Start the asynchronous request.
13    req.BeginGetResponse(new AsyncCallback(FinishWebRequest), Tuple
14        .Create(req,pt));
15    // set default state
16    lastState = (int)returnStates.timeout;
17    // wait for result
18    allDone.WaitOne(Timeout.Infinite);
19    // reset handle
20    allDone.Reset();
21 }
```

Переменная **req** является объектом класса **HttpWebRequest**, которая предназначена для отправки HTTP - запроса к серверу.

| Строка | Комментарий  |
|--------|--|
| 3      | Создание объекта <b>HttpWebRequest</b> с помощью метода <b>create</b> , в который передается адрес ресурса в виде строки.  |
| 4      | Указание метода запроса(POST, GET, HEAD...)  |
| 5      | Добавление информации об авторизации в заголовок запроса.  |
| 8      | Переменная <b>pt</b> является объектом перечисления <b>processType</b> . Использование данной переменной позволяет уменьшить количества повторяемого кода. Так в 10 строчке вызывается метод <b>FinishWebRequest</b> , который един для всех запросов типа GET, таким образом благодаря переменной <b>pt</b> определяются дальнейшие действия в методе <b>FinishWebRequest</b> . |
| 10     | Асинхронное получение ответа от сервера. В качестве объекта передается комбинация из запроса и типа запроса.   |
| 12     | Переменная <b>lastState</b> характеризует итоговой состояния какого-либо запроса. В данном случае ей присваивается значение ошибки по времени(ответ от сервера не был получен в установленный период). Значение данной переменной изменяется в зависимости от ответа сервера.  |
| 14     | Переменная <b>allDone</b> является объектом класса <b>ManualResetEvent</b> , которая необходима для контроля времени отведенного на прием ответа от сервера. По умолчанию значение <b>DefaultTimeout</b> равно 10 секундам.  |
| 16     | Возврат в исходное состояние, для последующих вызовов.   |

Таблица 3.2. Разбор кода из листинга 3.1

Для всех остальных GET - запросов, вышеприведенные действия

идентичны за исключением формирования самого запроса(строка 3 из листинга 3.1).

Метод FinishWebRequest класса engine, код которого приведен в листинге Д.2, проводит чтение JSON ответа сервера, а затем вызывает соответствующий метод, чтобы произвести синтаксический анализ("парсинг") результата. Такой анализ производится используя свободно распространяемую библиотеку **Json.NET**.

Рассмотрим такой анализ на примере получения текущего списка задач в системе.

Листинг 3.2. Метод parseTaskList класса engine.

```
1 private void parseTaskList(string input){  
2     if (input == null)  
3         return;  
4     Tasks.Data = new List<camundaTask>();  
5     var objects = JArray.Parse(input);  
6     foreach (JObject jo in objects){  
7         camundaTask temp = JsonConvert.DeserializeObject<camundaTask  
8             >(jo.ToString());  
9         Tasks.Data.Add(temp);  
10    }  
}
```

| Строка | Комментарий  |
|--------|--|
| 4      | Создание нового списка задач, элементами которого являются объекты класса <b>camundaTask</b> . |
| 5      | Создание массива объектов, путем синтаксического анализа строки <b>input</b> .                 |
| 6      | Создание объекта класса <b>camundaTask</b> путем десериализации ранее полученных объектов.     |
| 7      | Добавление новой задачи в общий список задач.  |

Таблица 3.3. Разбор кода из листинга 3.2

Подобный, достаточно простой, синтаксический анализ применим не ко всем методам. Так к примеру анализ строки с переменными задачи, потребовал внесения изменений(Приложение Б. Листинг Д.5) в процесс анализа строки.

Реализация POST - запросов схожа с GET - запросами, но с

некоторыми отличиями:

1. В момент формирования запроса, к нему также прикрепляются какие-либо данные, например текст нового комментария или иноформация о новой переменной задачи.
2. Ответ сервера не требует синтаксического анализа, требуется лишь обработка статус-кода ответа.

### 3.3.3. Графический интерфейс

При проектировании дизайна приложения, учитывалась требуемая функциональность приложения, так и сохранение простоты работы для пользователя.

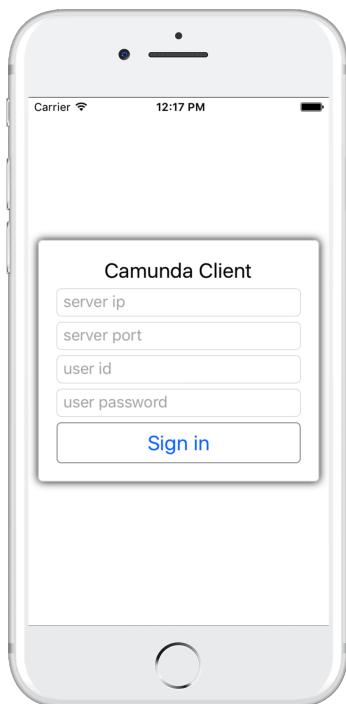


Рисунок 3.2. [IOS]Окно входа

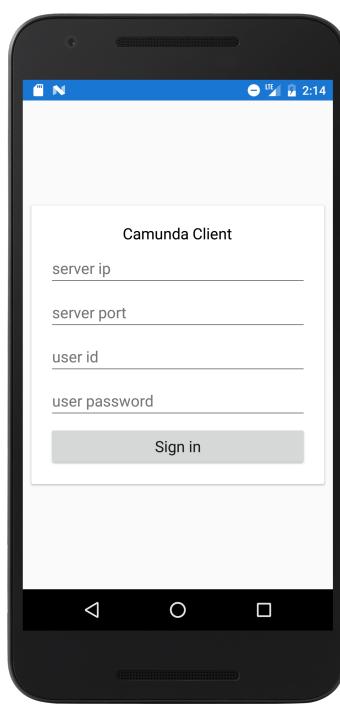


Рисунок 3.3. [Android]Окно входа

На рисунках 3.2 и 3.3 представлено окно для входа в систему Camunda. Общая концепция управления и дизайн одинаковы как для

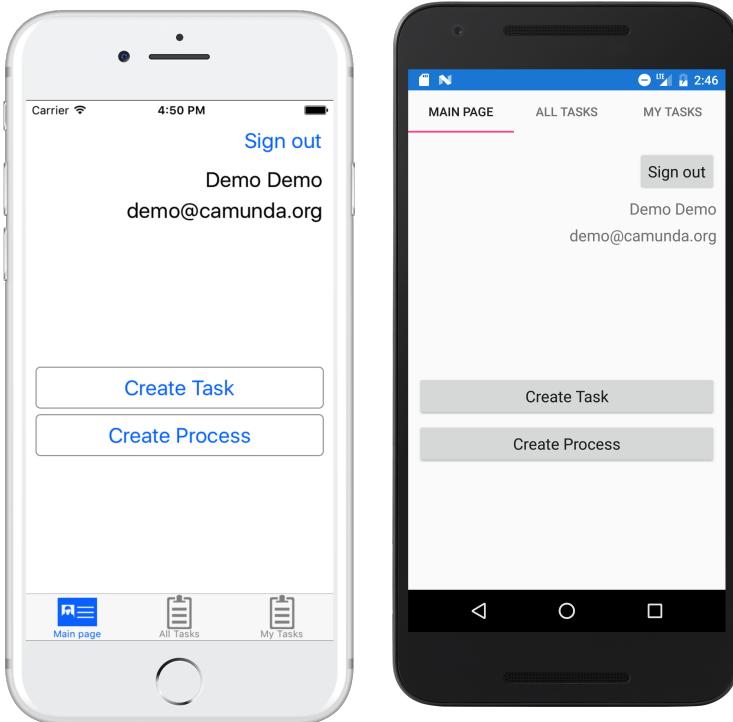


Рисунок 3.4. [IOS]Основное меню

Рисунок 3.5. [Android]Основное меню

IOS, так и для Android. Однако конечное отображение тех или иных элементов зависит от используемой мобильной платформы.

Наибольшие отличия имеет интерфейс основного меню, так на платформе IOS навигация между вкладками, расположена в нижней части экрана, в то время как у Android в верхней. Или же на Android имеется возможность перемещаться между вкладками путем их свайпа, а на IOS необходимо нажимать по требуемой вкладке.

Большее количество изображений графического интерфейса, представлены в приложении А.

### 3.4. Итоги

Было разработано кроссплатформенное мобильное приложение для платформ Android и IOS. Данное приложение реализует всю функциональность модуля **Tasklist** web-версии системы Camunda. Но в отличии от неё предоставляет всю информацию в понятном и доступном виде.

Использование кроссплатформенного фреймворка Xamarin.Forms показало что:

- Большая часть кода проекта будет являться кроссплатформенной.
- Конечный интерфейс приложения так или иначе будет отличаться на разных платформах.
- При работе с файловой системой устройства необходимо использовать платформозависимый код.

Xamarin.Forms весьма полезен в проектах, которые не работают с файловой системой и не используют платформозависимые возможности устройств, так как существенно сокращается время проектирования приложения. Иначе, лучше сразу проектировать для одной, конкретной платформы.

## 4. Тестирование системы

Тестирование было разделено на **backend**(внутренняя реализация) и **frontend**(внешнее представление) части.

### 4.1. Backend тестирование

В процессе разработки приложения, производилось поэтапное тестирование с целью выявления программных ошибок.

После завершения написания какого-либо класса, для него были написаны модульные тесты, тестирующие корректность его работы и отдельных методов. Наибольшее количество тестов имеет класс **engine**, регулирующий взаимодействия разработанного приложения и сервера с Camunda.

| Название теста                        | Описание   |
|---------------------------------------|--|
| Engine_Credentials                    | Проверка корректности данных сервера после создания объекта класса engine  |
| EngineAuthenticate                    | Аутентификация пользователя на сервере, с использованием корректных данных |
| EngineAuthenticate_No_Response        | Попытка подключения к не существующему адресу сервера                      |
| EngineAuthenticate_Bad_User           | Попытка аутентификации пользователя с неверным логином                     |
| EngineAuthenticate_Unreachable_Server | Попытка подключения к не существующему порту сервера                       |

|                               |  |
|-------------------------------|--|
| Engine_Get_Tasks              | Получение списка задач   |
| Engine_Get_Tasks_Not_Empty    | Проверка списка задач на наличие в нем задач                       |
| Engine_Get_All_Task_Variables | Проверка каждой задачи и процесса на наличие в них переменных      |
| Engine_Claim_Null_Task        | Попытка закрепления за текущим пользователем несуществующей задачи |

Таблица 4.1. Модульные тесты для класса engine

Большая часть тестов представляет из себя работу с REST - интерфейсом и проверкой результата подобного взаимодействия:

- Анализ кода ответа сервера, при симуляции различных ситуаций аутентификации;
- Проверка получаемых данных, на наличие в них требуемой информации.

## 4.2. Frontend тестирование

По мере разработки приложения, его отладка происходила на реальном устройстве **Samsung Galaxy Note 2** с версией Android 6.0.1. По завершению разработки какого-либо модуля и его успешной работы на данном устройстве, тестирование производилось в различных симуляторах операционных систем Android и IOS. Симуляторы отличались друг от друга:

- версией операционной системы;
- диагональю экрана;

- конечным устройством (в отличии от Android, для IOS необходимо выбрать реальную модель существующего мобильного устройства).

Для запуска симулятора на Android, необходимо установить необходимый SDK для необходимой версии Android и затем создать симулятор.

Для IOS запуск симулятора более затруднителен, так как дополнительно необходимо наличие операционной системы **Mac OS**, для запуска симулятора.

Для того чтобы вручную не тестировать графический интерфейс, были написаны соответствующие тесты.

Тестирование было реализовано с помощью фреймворка **Xamarin.UITest**, позволяющего тестировать графический интерфейс платформ Android и IOS. К его возможностям можно отнести:

1. Ввод текста в текстовые поля;
2. Нажатие по элементам графического интерфейса;
3. Реализация жестов.

Каждый такой тест можно разделить на 3 части:

1. Arrange - инициализация каких-либо параметров для выполнения теста;
2. Act - взаимодействие с приложением;
3. Assert - проверка результата на корректность.

Для того чтобы использовать данный фреймворк и начать писать тесты, необходимо так или иначе обращаться к элементам графического интерфейса. Для этого, для каждого элемента, может быть задано соответствующее свойство **AutomationId**, это позволяет напрямую обращаться к этому элементу.

Как альтернатива, в случае динамического содержания графического интерфейса, имеется возможность выполнить поиск требуемого элемента, по типу его класса, и затем обращаться к найденному объекту.

| Название теста | Описание  |
|----------------|---|
| AppLaunches    | Проверка на возможность запуска приложения  |
| authenticate   | Проверка на успешную аутентификацию, после ввода корректных данных пользователя и сервера |
| createTestTask | Проверка на успешное создание тестовой задачи   |
| claimTestTask  | Проверка на успешное закрепление за текущим пользователем, тестовой задачи                |
| addComment     | Проверка на успешное добавление комментария к тестовой задаче                             |
| comleteTask    | Проверка на успешное завершение, закрепленной за текущим пользователем, тестовой задачи   |

Таблица 4.2. Frontend тесты и их описания

Данные тесты могут быть реализованы в эмуляторе соответствующей ОС, или на реальном устройстве.

Для взаимодействия с платформозависимыми графическими элементами, при написании тестов, возможно произвести соответствующее разделение. В данной работе этого не потребовалось, тесты были успешно протестированы на платформах Android и IOS.

#### 4.2.1. Разбор теста

В качестве примера, была выбрана аутентификация пользователя. Необходимым для выполнения теста полям, были присвоены соответствующие значения, приведенные на рисунках 4.1 и 4.2.

Листинг 4.1. Тест **authenticate**

```

1 [Test]
2 public void authenticate()
3 {
4     app.EnterText("autoServerIp", trustedIp);
5     app.EnterText("autoServerPort", trustedPort);
6     app.EnterText("autoUserId", trustedUserId);
7     app.EnterText("autoUserPassword", trustedUserPassword);
8
9     app.Tap("autoSignIn");
10

```

```
11     app.WaitForElement(x => x.Marked("autoUserCred"), timeout:  
12         TimeSpan.FromSeconds(10));  
13     app.Repl();  
}
```

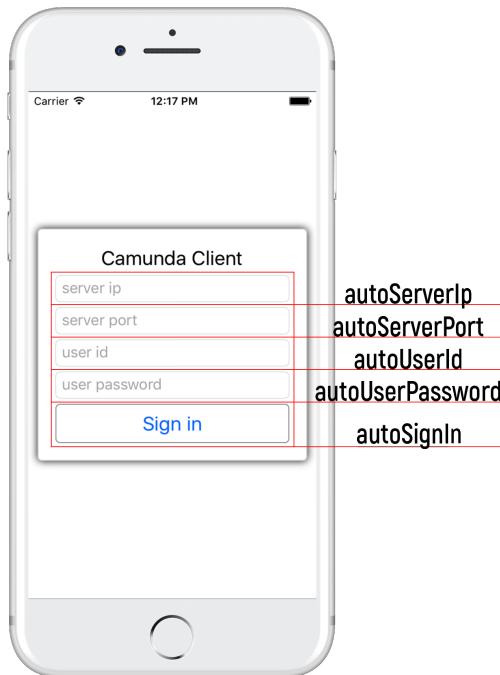


Рисунок 4.1. Окно входа, с помеченными элементами

Последовательность действий теста:

1. Ввод в поле **autoServerIp** константы **trustedIp**;
2. Ввод в поле **autoServerPort** константы **trustedPort**;
3. Ввод в поле **autoUserId** константы **trustedUserId**;
4. Ввод в поле **autoUserPassword** константы **trustedUserPassword**;
5. Нажатие по элементу(кнопке), помеченному как **autoSignIn**;

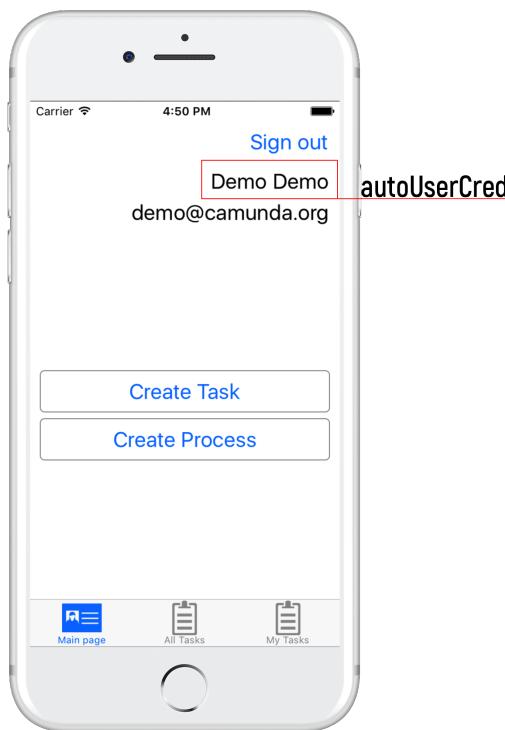


Рисунок 4.2. Главное меню, с помеченным элементом

6. 10 секунд ожидания элемента, помеченного как **autoUserCred**, в случае его не обнаружения, тест считается не пройденным;
7. Запуск(необязательное действие) утилиты REPL.

Листинг 4.2. Лог теста

```
1 ----- Начато выполнение тестов -----
2 NUnit VS Adapter 2.0.0.0 executing tests is started
3 Loading tests from C:\diplom\camundaClient\Camunda.UITests\bin\
   Release\Camunda.UITests.dll
4 Run started: C:\diplom\camundaClient\Camunda.UITests\bin\Release\
   Camunda.UITests.dll
5 Full log file: C:\Users\psaer\AppData\Local\Temp\uitest\log
   -2017-05-20_13-12-10-358.txt
6 Skipping IDE integration as important properties are configured. To
   force IDE integration, add .PreferIdeSettings() to ConfigureApp
```

```
7  Android test running Xamarin.UITest version: 1.3.8
8  Initializing Android app on device 4df188872a2aaf51 with apk: C:\diplom\camundaClient\camundaClient\camundaClient.Android\bin\Release\camundaClient.Android.apk
9  Skipping local screenshots. Can be enabled with EnableScreenshots()
   when configuring app.
10 Signing apk with Xamarin keystore.
11 Skipping installation: Already installed.
12 Using element matching Marked("autoServerIp").
13 Tapping coordinates [ 360, 484 ].
14 Using element matching Marked("autoServerPort").
15 Tapping coordinates [ 360, 587 ].
16 Using element matching Marked("autoUserId").
17 Tapping coordinates [ 360, 690 ].
18 Using element matching Marked("autoUserPassword").
19 Tapping coordinates [ 360, 793 ].
20 Using element matching Marked("autoSignIn").
21 Tapping coordinates [ 360, 894 ].
22 Waiting for element matching Marked("autoUserCred").
```

Тест был проведен на Android устройстве, о чём свидетельствует 8 строчка из листинга 4.2, далее в логе приведена описанная последовательность действий, с учетом координат экрана устройства, в которые были произведены нажатия.

После данных действий открывается окно утилиты REPL.

Утилита REPL(read-eval-print-loop) является инструментом для отлаживания тестов, в частности имеется возможность, представить графический интерфейс текущего состояния приложения в текстовом виде, что позволяет проанализировать всю иерархию элементов, представленных на экране устройства.

#### Листинг 4.3. Утилита REPL

```
1  Full log file: C:\Users\psaer\AppData\Local\Temp\uitest\log
   -2017-05-20_13-12-46-833.txt
2  Skipping IDE integration as important properties are configured. To
   force IDE integration, add .PreferIdeSettings() to ConfigureApp
.
3  Android test running Xamarin.UITest version: 1.3.8
4  Initializing Android app on device 4df188872a2aaf51.
5
6  App has been initialized to the 'app' variable.
7  Exit REPL with ctrl-c or see help for more commands.
8
9  >>> tree
10 [[object CalabashRootView] > ... > FrameLayout]
11   [FitWindowsFrameLayout] id: "action_bar_root"
12     [ContentFrameLayout > RelativeLayout] id: "content"
13       [View]
14         [PlatformRenderer]
15           [PageRenderer > ... > Platform_DefaultRenderer]
16             [LabelRenderer]
```

```

17      [FormsTextView] text: "Camunda Client"
18      [EntryRenderer] label: "autoServerIp_Container"
19          [EntryEditText] label: "autoServerIp", text: "
20              192.168.0.107"
21      [EntryRenderer] label: "autoServerPort_Container"
22          [EntryEditText] label: "autoServerPort", text: "8080"
23      [EntryRenderer] label: "autoUserId_Container"
24          [EntryEditText] label: "autoUserId", text: "demo"
25      [EntryRenderer] label: "autoUserPassword_Container"
26          [EntryEditText] label: "autoUserPassword", text: "demo"
27      [ButtonRenderer] label: "autoSignIn_Container"
28          [AppCompatButton] label: "autoSignIn", text: "Sign in"
29      [Platform_ModalContainer]
30          [View]
31          [TabbedPageRenderer]
32              [FormsViewPager > ... > Platform_DefaultRenderer] id: "
33                  NoResourceEntry -1"
34          [Platform_DefaultRenderer]
35              [ButtonRenderer]
36                  [AppCompatButton] text: "Sign out"
37          [LabelRenderer] label: "autoUserCred_Container"
38              [FormsTextView] label: "autoUserCred", text: "
39                  Demo Demo"
40          [LabelRenderer]
41              [FormsTextView] text: "demo@camunda.org"
42          [Platform_DefaultRenderer]
43              [ButtonRenderer] label: "autoCreateTask_Container"
44                  [AppCompatButton] label: "autoCreateTask", text:
45                      "Create Task"
46          [ButtonRenderer]
47              [AppCompatButton] text: "Create Process"
48          [TabLayout > TabLayout$SlidingTabStrip]
49              [TabLayout$TabView]
50                  [AppCompatTextView] text: "Main page"
51                  [TabLayout$TabView]
52                      [AppCompatTextView] text: "All Tasks"
53                  [TabLayout$TabView]
54                      [AppCompatTextView] text: "My Tasks"

```

В листинге 4.3 была использована команда **tree**, что позволило посмотреть всю иерархию элементов. Помимо команды **tree**, имеются следующие команды:

- **Flash** - Подробный вывод всей информации по элементам заданного класса;
- **EnterText** - Ввод текста в заданный элемент;
- **Tap** - нажатие по заданному элементу;
- **Copy** - скопировать текст лога.

## 4.3. Итоги

Тестирование какого-либо проекта является его неотъемлемой частью. Разделение тестирования на backend и frontend части, в данном случае позволяет в полной мере протестировать приложение. Так если backend тесты, после изменений проекта продолжают успешно выполняться, то нельзя однозначно сказать сможет ли пользователь корректно взаимодействовать с приложением. Поэтому необходимы frontend тесты, которые по шагам выполняют поставленные задачи.

Встроенные в Xamarin возможности по тестированию позволяют:

- Пронаблюдать какие действия происходят в ходе теста;
- Просмотреть всю иерархию элементов на экране устройства, и при необходимости вывести подробное описание каждого элемента;
- По завершению теста продолжить тест в ручном режиме.

Все это позволяет ускорить тестирование, а также внести понимание того, что происходит на экране устройства.

# ЗАКЛЮЧЕНИЕ

В результате выполнения работы были исследованы системы по автоматизации бизнес процессов. Исследование показало, что крупным компаниям разумнее использовать свободное ПО, так как стоимость лицензии, коммерческих систем, напрямую зависит от количества пользователей. Среди свободно-распространяемых систем, наиболее перспективной является **Camunda BPM**, однако у данной системы не оказалось мобильного клиента, что характерно для всех коммерческих систем.

У системы **Camunda BPM** имеется несколько модулей, задачей данной работы был перенос всех возможностей модуля **Tasklist** на мобильное устройство в виде приложения - клиента. Стоит учесть, что для большего охвата пользователей, конечное мобильное приложение должно быть кроссплатформенным. Наиболее оптимальным, с точки зрения "цена/качество", был выбран фреймворк Xamarin. Его использование позволило существенно сократить количество требуемого для написания кода, в тоже время сохранив весь потенциал возможностей конечной платформы.

В ходе тестирования приложения, был применен инструмент Xamarin под названием REPL. Его использование позволило проанализировать всю иерархию элементов в ходе тестирования графического интерфейса. Также с его помощью имеется возможность произвести тестирование в ручном режиме. Данный инструмент будет особенно полезен в интерфейсах, с большим количеством элементов.

Разработанное кроссплатформенное мобильное приложение в полной мере обеспечивает функциональность модуля **Tasklist** системы **Camunda BPM**. Таким образом имеется возможность автоматизировать работу с данной системой, через мобильные устройства.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. BPM (управленческая концепция) [Электронный ресурс], Википедия — свободная энциклопедии. — URL: [https://ru.wikipedia.org/wiki/BPM\\_\(управленческая\\_концепция\)](https://ru.wikipedia.org/wiki/BPM_(управленческая_концепция)).
2. Micro-Benchmarking BPMN 2.0 Workflow Management Systems with Workflow Patterns / Marigianna Skouradaki, Vincenzo Ferme, Cesare Pautasso и др. // Advanced Information Systems Engineering: 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings / Под ред. Selmin Nurcan, Pnina Soffer, Marko Bajec, Johann Eder. — Cham : Springer International Publishing, 2016. — С. 67–82. — ISBN: 978-3-319-39696-5. — URL: [http://dx.doi.org/10.1007/978-3-319-39696-5\\_5](http://dx.doi.org/10.1007/978-3-319-39696-5_5).
3. Документация REST интерфейса бизнес системы Camunda [Электронный ресурс], Camunda Docs. — URL: <https://docs.camunda.org/manual/latest/reference/rest/> (дата обращения: 12.05.2017).
4. Обзор кросс-платформенных решений для разработки мобильных приложений [Электронный ресурс], Хабрахабр. — URL: <https://habrahabr.ru/post/319348/> (дата обращения: 02.05.2017).
5. Документация по тестированию графического интерфейса в Xamarin [Электронный ресурс], Xamarin. — URL: <https://developer.xamarin.com/guides/testcloud/uitest/intro-to-uitest/> (дата обращения: 20.05.2017).
6. Документация утилиты REPL фреймворка Xamarin [Электронный ресурс], Xamarin. — URL: <https://developer.xamarin.com/guides/testcloud/uitest/working-with/repl/> (дата обращения: 20.05.2017).
7. Документация по модулю Xamarin.Forms фреймворка Xamarin [Электронный ресурс], Xamarin. — URL: <https://developer.xamarin.com/guides/xamarin-forms/> (дата обращения: 21.05.2017).
8. Документация библиотеки Json.NET [Электронный ресурс], Newtonsoft. — URL: <http://www.newtonsoft.com/json/help/html/Introduction.htm> (дата обращения: 15.04.2017).
9. Hermes Dan. Xamarin Mobile Application Development: Cross-

- Platform C# and Xamarin. Forms Fundamentals. — 1 изд. — Berkely, CA, USA : Apress, 2015. — ISBN: 1484202155, 9781484202159.
10. Smith William. Learning Xamarin Studio. — Packt Publishing, 2014. — ISBN: 1783550813, 9781783550814.

# ПРИЛОЖЕНИЕ А

## Дополнительные скриншоты

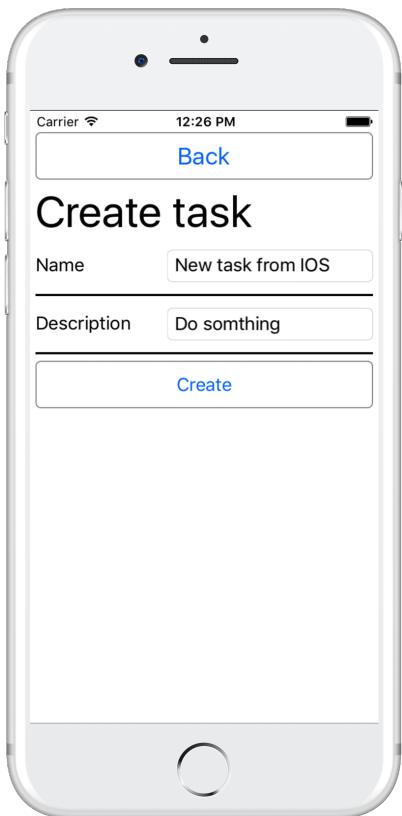


Рисунок А.1. [IOS]Создание новой задачи

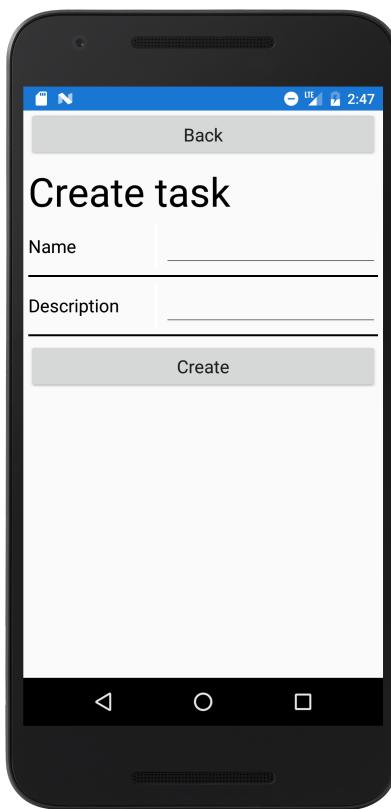


Рисунок А.2. [Android]Создание новой задачи



Рисунок А.3. [IOS] Выбор нового процесса

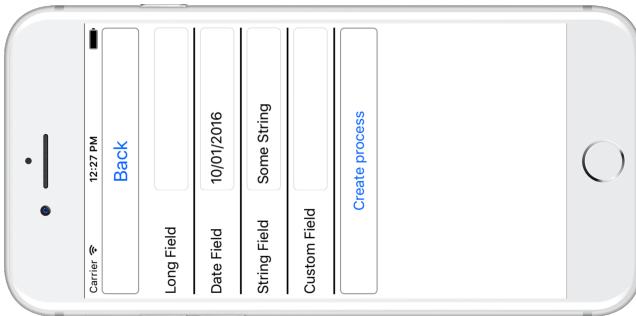


Рисунок А.4. [IOS] Ввод начальных переменных, при создании нового процесса



Рисунок А.5. [IOS] Список текущих задач и процессов

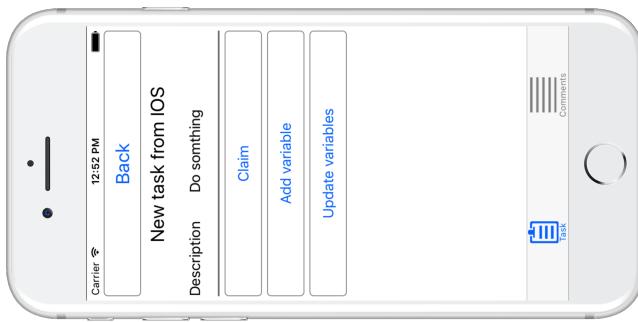


Рисунок А.6. [IOS]Просмотр задачи

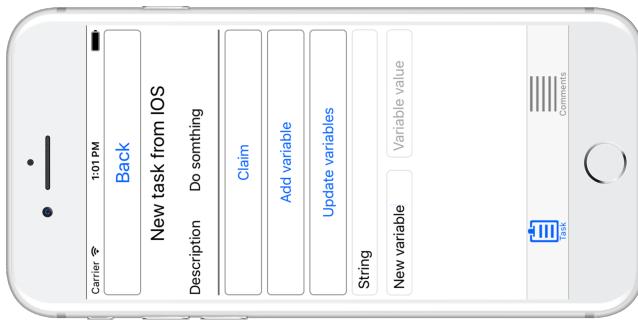


Рисунок А.7. [IOS]Добавление дополнительных переменных к задаче



Рисунок А.8. [IOS]Окно просмотра и добавления комментариев

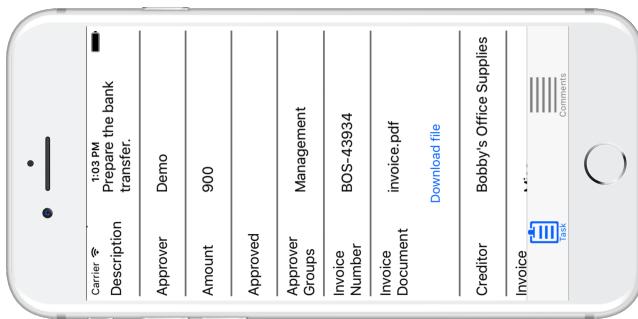


Рисунок А.9. [IOS]Просмотр процесса

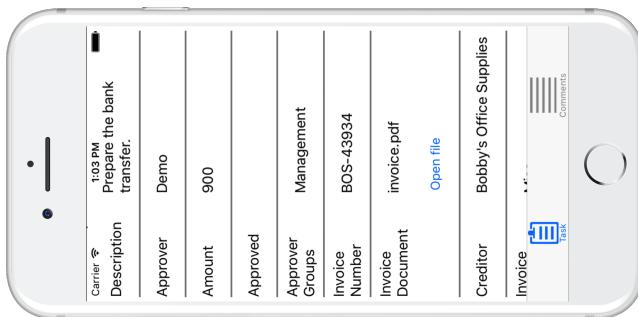


Рисунок А.10. [IOS]После скачивания файла, кнопка скачивание теперь открывает скачанный файл



Рисунок А.11. [IOS]Открытый pdf файл

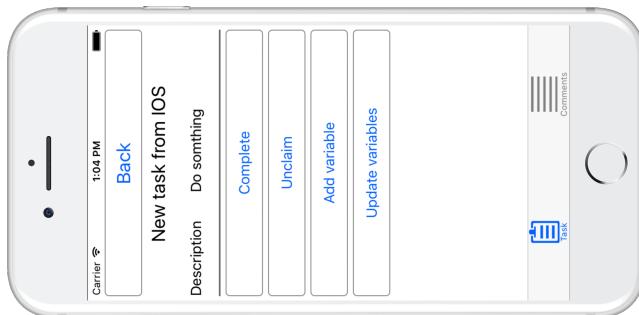


Рисунок А.12. [iOS] Если задача или процесс закреплены за пользователем, то добавляется возможность завершения

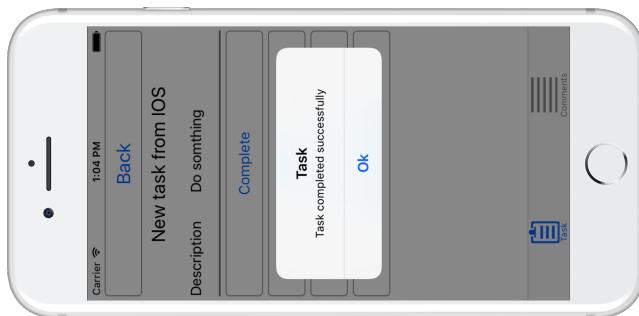


Рисунок А.13. [iOS] Информирование об успешном завершении

## ПРИЛОЖЕНИЕ Б

Кроссплатформенное мобильное приложение - клиент системы управления бизнес процессами Camunda.

Техническое задание.

1 лист

## **Введение**

Наименование программы: «Кроссплатформенное мобильное приложение - клиент системы управления бизнес процессами Camunda». Данное средство предназначено для управления бизнес процессами системы Camunda, через мобильное устройство.

## **Назначение разработки**

Обеспечение скорости и удобства сотрудников в системе Camunda BPM без привязки к рабочему месту в офисе.

## **Требование к программе или программному изделию**

### *Требования к функциональным характеристикам*

Разработанное приложение должно поддерживать все возможности модуля Tasklist web-версии системы, в частности:

- Создание произвольных задач или старт предварительно определенных бизнес-процессов;
- Получение списка текущих бизнес-процессов или задач, а также их состояния и возможностью просмотром более детальной информации;
- Разграничение задач по пользователям;
- Изменение статуса и комментирования бизнес-процессов или задач.

Производить аутентификацию в системе, с помощью логина и пароля какого-либо пользователя системы. Возможность задания адреса и порта сервера с развернутой системой Camunda BPM.

### *Требования к составу и параметрам технических средств*

Разработанное кроссплатформенное мобильное приложение может функционировать на аппаратной платформе следующих мобильных операционных системах:

- Android (версия 4.0.3 и старше)
- IOS (версия 6.1 и старше)

### *Требования к информационной и программной совместимости*

Программа должна быть написана на языке C# с использованием фреймворка Xamarin, и взаимодействовать с сервером Camunda через REST-интерфейс.

## ПРИЛОЖЕНИЕ В

Кроссплатформенное мобильное приложение - клиент системы  
управления бизнес процессами Camunda.

Описание программы.

1 лист

## **Общие сведения**

Разработанное кроссплатформенное мобильное приложение, реализует всю функциональность модуля Tasklist web-версии системы Camunda. Это позволяет автоматизировать работу с бизнес процессами через мобильное устройство.

## **Используемые технические средства**

Для функционирования приложения, необходимо подключение к сети в которой находится сервер с Camunda, а также одна из следующих мобильных операционных систем:

- Android (версия 4.0.3 и старше);
- IOS (версия 6.1 и старше).

## **Входные данные**

Входными данными являются сетевая конфигурация:

1. адрес сервера;
2. порт сервера.

А также данные пользователя, необходимые для аутентификации:

1. логин пользователя;
2. пароль пользователя.

## ПРИЛОЖЕНИЕ Г

Кроссплатформенное мобильное приложение - клиент системы управления бизнес процессами Camunda.

Программа и методика испытаний.

3 листа

## **Объект испытаний**

Наименование программы – объекта испытаний: «Кроссплатформенное мобильное приложение - клиент системы управления бизнес процессами Camunda».

## **Цель испытаний**

Целью проведения испытаний является проверка соответствия функциональных характеристик средства требованиям, указанным в техническом задании.

## **Требования к программе**

Разработанное средство должно функционировать на мобильных операционных систем:

- Android (версия 4.0.3 и старше);
- IOS (версия 6.1 и старше).

А также подключаться к заданному серверу с Camunda, и удалённо производить все действия модуля Tasklist с бизнес процессами.

## **Средства и порядок испытаний**

Программа должна выполняться на эмуляторах мобильных операционных систем Android(версия 4.0.3 и старше) и IOS(версия 6.1 и старше) или на физических устройствах с соответствующей версией ОС. Эмулятор или устройство, на котором проходит тестирование, должно находится в одной сети с компьютером на котором развернута бизнес система Camunda. Для проведения испытаний был собран тестовый стенд, состоящий из трех компьютеров, находящихся в одной сети.

Компьютер с эмулятором Android:

### 1. Оборудование:

- Intel Core i5 4210U;
- Оперативная память – 8 Гб;
- Жёсткий диск – 250 Гб;
- Кarta сетевого интерфейса(NIC);
- Стандартный монитор;
- Клавиатура;
- Мышь.

### 2. Операционная система:

- Windows 10.

3. Установленное ПО:

- Android SDK(API 15 - API 25);
- Фреймворк Xamarin;
- Microsoft Visual Studio 2015.

Компьютер с эмулятором IOS:

1. Оборудование:

- Intel Core i5 5250U;
- Оперативная память – 4 Гб;
- Жёсткий диск – 128 Гб;
- Кarta сетевого интерфейса(NIC);
- Стандартный монитор;
- Клавиатура;
- Мышь.

2. Операционная система:

- Mac OS X v10.10 Yosemite.

3. Установленное ПО:

- IOS SDK;
- Фреймворк Xamarin;
- Xamarin Studio.

Компьютер с развернутой Camunda BPM:

1. Оборудование:

- AMD FX 8120;
- Оперативная память – 6 Гб;
- Жёсткий диск – 500 Гб;
- Кара сетевого интерфейса(NIC);
- Стандартный монитор;

- Клавиатура;
  - Мышь.
2. Операционная система:
    - Linux Mint 18.1 Serena.
  3. Установленное ПО:
    - Camunda BPM.

**Порядок проведения испытаний:**

1. запуск сервера с Camunda BPM;
2. проверка на нахождение в одной сети всех трех компьютеров;
3. выбор конфигурации эмуляторов Android и IOS;
4. запуск эмуляторов Android и IOS;
5. запуск программного средства;
6. проверка на соответствие функциональных характеристик, указанных в техническом задании;
7. анализ проведённых испытаний.

## ПРИЛОЖЕНИЕ Д

Кроссплатформенное мобильное приложение - клиент системы  
управления бизнес процессами Camunda.

Фрагменты кода программы.

22 листа

Листинг Д.1. createTask.xaml.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Linq;
5  using System.Runtime.CompilerServices;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Windows.Input;
9
10 using Xamarin.Forms;
11 using Xamarin.Forms.Xaml;
12
13 namespace camundaClient
14 {
15
16     [XamlCompilation(XamlCompilationOptions.Compile)]
17     public partial class createTask : ContentPage
18     {
19         private Entry entryName;
20         private Entry entryDescription;
21         private engine engine;
22
23         public createTask(engine engine)
24         {
25             InitializeComponent();
26             createTaskView.Padding = new Thickness(5, Device.OnPlatform(20, 0, 0), 5, 0);
27             this.engine = engine;
28             btnBack.Clicked += (s, e) =>{ base.OnBackButtonPressed();
29             };
30
31             addSimpleLine("Name", 1);
32             addSimpleLine("Description", 2);
33             addButton();
34         }
35
36         private async Task createMyTask()
37         {
38             await Task.Run(() => engine.createTask(entryName.Text,
39                 entryDescription.Text));
40             if (engine.lastState == 0)
41                 notify("Task", "Task created successfully");
42             else
43                 notify("Task", "Can't create task");
44         }
45         private void notify(string header, string msg)
46         {
47             DisplayAlert(header, msg, "Ok");
48         }
49         private void addButton()
50         {
51             Button btnAdd = new Button
52             {
53                 Text = "Create",
54                 BorderColor = Color.Gray,
55                 BorderWidth = 1,
```

```

54         FontSize = Device.GetNamedSize(NamedSize.Medium,
55                                         typeof(Label))
56     };
57     btnAdd.Clicked += (s, e) =>{ createMyTask(); };
58
59     createTaskView.Children.Add(btnAdd);
60 }
61 private void addSimpleLine(string name, int type){
62     StackLayout cell = new StackLayout{
63         Orientation = StackOrientation.Horizontal
64     };
65     Grid tempGrid = new Grid();
66     tempGrid.RowDefinitions.Add(new RowDefinition { Height =
67         GridLength.Auto });
68     tempGrid.ColumnDefinitions.Add(new ColumnDefinition {
69         Width = new GridLength(45, GridUnitType.Star) });
70     tempGrid.ColumnDefinitions.Add(new ColumnDefinition {
71         Width = new GridLength(1, GridUnitType.Star) });
72     tempGrid.ColumnDefinitions.Add(new ColumnDefinition {
73         Width = new GridLength(80, GridUnitType.Star) );
74
75     tempGrid.Children.Add(
76         new Label{
77             Text = name,
78             TextColor = Xamarin.Forms.Color.Black,
79             FontSize = Device.GetNamedSize(NamedSize.Medium,
80                                             typeof(Label)),
81             LineBreakMode = LineBreakMode.WordWrap,
82             VerticalOptions = LayoutOptions.Center,
83             HorizontalOptions = LayoutOptions.StartAndExpand
84         }, 0, 0
85     );
86     tempGrid.Children.Add(new BoxView{
87         Color = Color.White
88     }, 1, 0);
89
90     if (type == 1)
91     {
92         entryName = new Entry
93     {
94         TextColor = Xamarin.Forms.Color.Black,
95         FontSize = Device.GetNamedSize(NamedSize.Medium,
96                                         typeof(Label)),
97         VerticalOptions = LayoutOptions.Center
98     };
99
100    tempGrid.Children.Add(
101        entryName, 2, 0
102    );
103 }
104 else
105 {
106     entryDescription = new Entry
107     {
108         TextColor = Xamarin.Forms.Color.Black,
109         FontSize = Device.GetNamedSize(NamedSize.Medium,
110                                         typeof(Label)),
111

```

```

104             VerticalOptions = LayoutOptions.Center
105         };
106
107         tempGrid.Children.Add(
108             entryDescription, 2, 0
109         );
110     }
111
112     createTaskView.Children.Add(tempGrid);
113     createTaskView.Children.Add(new BoxView() { Color =
114         Color.Black, WidthRequest = 100, HeightRequest = 2
115     });
116 }
117
}

```

Листинг Д.2. engine.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Net;
7  using System.IO;
8  using System.Threading;
9  using Newtonsoft.Json;
10 using Newtonsoft.Json.Linq;
11
12 namespace camundaClient
13 {
14     enum returnStates{
15         allIsOk = 0x0,
16         timeout = 0x1,
17         unauthorized = 0x2,
18         unreachable = 0x3,
19         unknownError = 0x4
20     }
21     enum processType{
22         userInfo=0x0,
23         taskList=0x1,
24         variables=0x2,
25         file=0x3,
26         createTask=0x4,
27         comment=0x5,
28         processes=0x6,
29         processesVariables=0x7
30     }
31
32
33
34     public class engine
35     {
36         //for long task control
37         private static ManualResetEvent allDone = new
38             ManualResetEvent(false);

```

```

38     private const int DefaultTimeout = 10000; // 10 seconds timeout
39
40     public string serverIp { get; set; }
41     public string serverPort { get; set; }
42     public camundaProcs procs { get; set; }
43
44     private HttpWebRequest req;
45
46     private camundaUser user;
47     private camundaTasks tasks;
48
49     public camundaUser User{
50         get{ return user;}
51         set{user = value;}
52     }
53
54     public camundaTasks Tasks{
55         get{ return tasks; }
56         set{ tasks = value; }
57     }
58
59     public int lastState;
60     public int lastElement;
61     public int lastProcKeyId;
62
63     //for debug
64     public string test;
65
66     public string getProcCaseInstId()
67     {
68         if (String.IsNullOrEmpty(Tasks.Data.ElementAt(
69             lastElement).ProcessInstanceId))
70             return Tasks.Data.ElementAt(lastElement).
71                 CaseInstanceId;
72         else
73             return Tasks.Data.ElementAt(lastElement).
74                 ProcessInstanceId;
75     }
76
77     public bool isProcInstance()
78     {
79         if (String.IsNullOrEmpty(Tasks.Data.ElementAt(
80             lastElement).ProcessInstanceId))
81             return false;
82         else
83             return true;
84     }
85
86     public engine(string serverIp, string serverPort,
87                   camundaUser user)
88     {
89         this.serverIp = serverIp;
90         this.serverPort = serverPort;
91         this.User = user;
92     }
93
94     public async Task createProc(List<taskPageItem> items)
95     {
96         req = null;

```

```

91     req = (HttpWebRequest)WebRequest.Create(@"http://" +
92         serverIp +
93         ":" + serverPort + "/engine-rest/process-definition/
94         key/" + procs.data.ElementAt(lastProcKeyId).key
95         + "/start");
96     req.Method = "POST";
97     req.Headers["Authorization"] = "Basic " + Convert.
98         ToBase64String(
99             Encoding.UTF8.GetBytes(User.Id.ToString() + ":" +
100                 User.Password.ToString())));
101    req.ContentType = "application/json";
102
103    string postData = "{ \"variables\":{ ";
104    foreach (taskPageItem item in items){
105        if (item.type == valueType.simpleString)
106            postData += "\"" + item.name + "\":{\\"value\":\""
107                + item.ed.Text + "\",\"type\":\"String"
108                + "\"},";
109        else if (item.type == valueType.d_number)
110            postData += "\"" + item.name + "\":{\\"value\":\""
111                + item.ed.Text + "\",\"type\":\"Double"
112                + "\"},";
113    }
114    postData = postData.Substring(0, postData.Length - 1);
115    postData += "}";
116
117    var stream = await req.GetRequestStreamAsync();
118    using (var writer = new StreamWriter(stream))
119    {
120        writer.Write(postData);
121        writer.Flush();
122        writer.Dispose();
123    }
124    getPostResponse(req);
125
126    // set default state
127    lastState = (int)returnStates.timeout;
128    // waiting for result
129    allDone.WaitOne(DefaultTimeout);
130    // reset handle
131    allDone.Reset();
132
133
134 }
```

public async Task updateTask(string taskId, List<taskPageVar> items){

```

135     req = null;
136     req = (HttpWebRequest)WebRequest.Create(@"http://" +
137         serverIp +
138         ":" + serverPort + "/engine-rest/task/" + taskId + "/
139         /variables");
140     req.Method = "POST";
141     req.Headers["Authorization"] = "Basic " + Convert.
142         ToBase64String(
143             Encoding.UTF8.GetBytes(User.Id.ToString() + ":" +
144                 User.Password.ToString())));
145     req.ContentType = "application/json";
146
147     string postData = "{ \"modifications\":{ ";
```

```

135     foreach (taskPageVar item in items){
136         if(item.type==valueType.simpleString)
137             postData += "\"" + item.varName.Text + "\" : {" +
138                 value :"\"" + item.varValue.Text + "\",\"type" +
139                 "\" : \"String\"},";
140     }
141     postData = postData.Substring(0, postData.Length - 1);
142     postData += "}";
143 //test = postData;
144
145     var stream = await req.GetRequestStreamAsync();
146     using (var writer = new StreamWriter(stream))
147     {
148         writer.Write(postData);
149         writer.Flush();
150         writer.Dispose();
151     }
152     getPostResponse(req);
153
154     // set default state
155     lastState = (int)returnStates.timeout;
156     // waiting for result
157     allDone.WaitOne(DefaultTimeout);
158     // reset handle
159     allDone.Reset();
160 }
161
162 public async Task addComment(string taskId, string comment)
163 {
164     req = null;
165     req = (HttpWebRequest)WebRequest.Create(@"http://" +
166           serverIp +
167           ":" + serverPort + "/engine-rest/task/" + taskId + "/
168           /comment/create");
169     req.Method = "POST";
170     req.Headers["Authorization"] = "Basic " + Convert.
171           ToBase64String(
172           Encoding.UTF8.GetBytes(User.Id.ToString() + ":" +
173               User.Password.ToString()));
174     req.ContentType = "application/json";
175
176     string postData = "{\"userId\":\"" + user.Id + "\",\""
177           message :\'" + comment+"\")";
178     var stream = await req.GetRequestStreamAsync();
179     using (var writer = new StreamWriter(stream))
180     {
181         writer.Write(postData);
182         writer.Flush();
183         writer.Dispose();
184     }
185
186     getPostResponse(req);
187
188     // set default state

```

```

184         lastState = (int) returnStates.timeout;
185         // waiting for result
186         allDone.WaitOne(DefaultTimeout);
187         // reset handle
188         allDone.Reset();
189     }
190     public async Task completeTask(string taskId)
191     {
192         req = null;
193         req = (HttpWebRequest)WebRequest.Create(@"http://" +
194             serverIp +
195             ":" + serverPort + "/engine-rest/task/" + taskId + " +
196             "/complete");
197         req.Method = "POST";
198         req.Headers["Authorization"] = "Basic " + Convert.
199             ToBase64String(
200                 Encoding.UTF8.GetBytes(User.Id.ToString() + ":" +
201                     User.Password.ToString())));
202         req.ContentType = "application/json";
203
204         string postData = "{}";
205         var stream = await req.GetRequestStreamAsync();
206         using (var writer = new StreamWriter(stream))
207         {
208             writer.Write(postData);
209             writer.Flush();
210             writer.Dispose();
211         }
212         getPostResponse(req);
213
214         // set default state
215         lastState = (int) returnStates.timeout;
216         // waiting for result
217         allDone.WaitOne(DefaultTimeout);
218         // reset handle
219         allDone.Reset();
220     }
221     public async Task unclaimTask(string taskId)
222     {
223         req = null;
224         req = (HttpWebRequest)WebRequest.Create(@"http://" +
225             serverIp +
226             ":" + serverPort + "/engine-rest/task/" + taskId + " +
227             "/unclaim");
228         req.Method = "POST";
229         req.Headers["Authorization"] = "Basic " + Convert.
230             ToBase64String(
231                 Encoding.UTF8.GetBytes(User.Id.ToString() + ":" +
232                     User.Password.ToString())));
233         req.ContentType = "application/json";
234
235         var stream = await req.GetRequestStreamAsync();
236         using (var writer = new StreamWriter(stream))
237         {
238             writer.Flush();
239             writer.Dispose();
240         }
241         getPostResponse(req);

```

```

234         // set default state
235         lastState = (int)returnStates.timeout;
236         // waiting for result
237         allDone.WaitOne(DefaultTimeout);
238         // reset handle
239         allDone.Reset();
240     }
241
242
243     public async Task claimTask(string taskId)
244     {
245         req = null;
246         req = (HttpWebRequest)WebRequest.Create(@"http://" +
247             serverIp +
248             ":" + serverPort + "/engine-rest/task/" + taskId + "/claim");
249         req.Method = "POST";
250         req.Headers["Authorization"] = "Basic " + Convert.
251             ToBase64String(
252                 Encoding.UTF8.GetBytes(User.Id.ToString() + ":" +
253                     User.Password.ToString()));
254         req.ContentType = "application/json";
255
256         string postData = "{\"userId\":\"" + user.Id + "\"}";
257         var stream = await req.GetRequestStreamAsync();
258         using (var writer = new StreamWriter(stream))
259         {
260             writer.Write(postData);
261             writer.Flush();
262             writer.Dispose();
263         }
264         getPostResponse(req);
265
266         // set default state
267         lastState = (int)returnStates.timeout;
268         // waiting for result
269         allDone.WaitOne(DefaultTimeout);
270         // reset handle
271         allDone.Reset();
272     }
273
274     public async Task createTask(string inputName, string
275         inputDescription){
276         req = null;
277         req = (HttpWebRequest)WebRequest.Create(@"http://" +
278             serverIp +
279             ":" + serverPort + "/engine-rest/task/create");
280         req.Method = "POST";
281         req.Headers["Authorization"] = "Basic " + Convert.
282             ToBase64String(
283                 Encoding.UTF8.GetBytes(User.Id.ToString() + ":" +
284                     User.Password.ToString()));
285         req.ContentType = "application/json";
286
287         string postData = "{\"name\":\"" + inputName + "\", \"
288             description\":\"" + inputDescription + "\"}";
289         var stream = await req.GetRequestStreamAsync();
290         using (var writer = new StreamWriter(stream))
291         {
292             writer.Write(postData);
293             writer.Flush();

```

```

283             writer.Dispose();
284         }
285         getPostResponse(req);
286
287         // set default state
288         lastState = (int)returnStates.timeout;
289         // waiting for result
290         allDone.WaitOne(DefaultTimeout);
291         // reset handle
292         allDone.Reset();
293     }
294     private async void getPostResponse(HttpWebRequest request){
295         try{
296             var response = (HttpWebResponse)await req.
297                 GetResponseAsync();
298
299             if (response.StatusCode == HttpStatusCode.NoContent
300                 || response.StatusCode== HttpStatusCode.OK){
301                 lastState = (int)returnStates.allIsOk;
302             }
303             else{
304                 lastState = (int)returnStates.unknownError;
305             }
306         }
307         catch(Exception ex)
308         {
309             lastState = (int)returnStates.unknownError;
310         }
311         finally
312         {
313             allDone.Set();
314         }
315     }
316     public void getVariables(string _taskId)
317     {
318         for (int i = 0; i < Tasks.Data.Count(); i++)
319         {
320             if ( Tasks.Data.ElementAt(i).Id==_taskId){
321                 lastElement = i;
322                 req = null;
323
324                 req = (HttpWebRequest)WebRequest.Create(@"http
325                     ://"+ serverIp +
326                     ":" + serverPort + "/engine-rest/task/" +
327                     Tasks.Data.ElementAt(i).Id +
328                     "/form-variables");
329
329                 req.Method = "GET";
330                 req.Headers["Authorization"] = "Basic " +
331                     Convert.ToBase64String(
332                         Encoding.UTF8.GetBytes(User.Id.ToString() +
333                         ":" + User.Password.ToString()));
334
334                 // Adding type of response what are we waiting for
335                 processType pt = processType.variables;
336                 // Start the asynchronous request.

```

```

335             req.BeginGetResponse(new AsyncCallback(
336                 FinishWebRequest), Tuple.Create(req, pt));
337             // set default state
338             lastState = (int)returnStates.timeout;
339             // waitig for result
340             allDone.WaitOne(DefaultTimeout);
341             // reset handle
342             allDone.Reset();
343         }
344     }
345
346     public void getComments(string taskId)
347     {
348         for (int i = 0; i < Tasks.Data.Count(); i++)
349         {
350             if (Tasks.Data.ElementAt(i).Id == taskId)
351             {
352                 lastElement = i;
353                 req = null;
354
355                 req = (HttpWebRequest)WebRequest.Create(@"http
356                     ://" + serverIp +
357                     ":" + serverPort + "/engine-rest/task/" +
358                     Tasks.Data.ElementAt(i).Id +
359                     "/comment");
360
361                 req.Method = "GET";
362                 req.Headers["Authorization"] = "Basic " +
363                     Convert.ToBase64String(
364                         Encoding.UTF8.GetBytes(User.Id.ToString() +
365                         ":" + User.Password.ToString()));
366
367                 // Adding type of response what are we waiting for
368                 processType pt = processType.comment;
369                 // Start the asynchronous request.
370                 req.BeginGetResponse(new AsyncCallback(
371                     FinishWebRequest), Tuple.Create(req, pt));
372                 //req.BeginGetResponse(new
373                 //    AsyncCallback(FinishWebRequest), req);
374                 // set default state
375                 lastState = (int)returnStates.timeout;
376                 // waitig for result
377                 allDone.WaitOne(DefaultTimeout);
378                 // reset handle
379                 allDone.Reset();
380             }
381         }
382     }
383
384     public void getProcDefVars(string procDefKey)
385     {
386         req = null;
387         req = (HttpWebRequest)WebRequest.Create(@"http://" +
388             serverIp +
389             ":" + serverPort + "/engine-rest/process-definition/
390             key/" + procDefKey + "/form-variables");
391         req.Method = "GET";
392         req.Headers["Authorization"] = "Basic " + Convert.

```

```

384     ToBase64String(
385     Encoding.UTF8.GetBytes(User.Id.ToString() + ":" +
386     User.Password.ToString()));
387
388     for (int i = 0; i < procs.data.Count; i++)
389     if (procs.data.ElementAt(i).key == procDefKey)
390         lastProcKeyId = i;
391
392     // Adding type of response what are we waiting for
393     processType pt = processType.processesVariables;
394     // Start the asynchronous request.
395     req.BeginGetResponse(new AsyncCallback(FinishWebRequest)
396     , Tuple.Create(req, pt));
397     //req.BeginGetResponse(new AsyncCallback(FinishWebRequest),
398     //req);
399
400     // set default state
401     lastState = (int)returnStates.timeout;
402     // waiting for result
403     allDone.WaitOne(DefaultTimeout);
404     // reset handle
405     allDone.Reset();
406 }
407
408 public void getProcs()
409 {
410     req = null;
411     req = (HttpWebRequest)WebRequest.Create(@"http://" +
412     serverIp +
413     ":" + serverPort + "/engine-rest/process-definition"
414     );
415     req.Method = "GET";
416     req.Headers["Authorization"] = "Basic " + Convert.
417     ToBase64String(
418     Encoding.UTF8.GetBytes(User.Id.ToString() + ":" +
419     User.Password.ToString()));
420
421     procs = new camundaProcs();
422
423     // Adding type of response what are we waiting for
424     processType pt = processType.processes;
425     // Start the asynchronous request.
426     req.BeginGetResponse(new AsyncCallback(FinishWebRequest)
427     , Tuple.Create(req, pt));
428     //req.BeginGetResponse(new AsyncCallback(FinishWebRequest),
429     //req);
430
431     // set default state
432     lastState = (int)returnStates.timeout;
433     // waiting for result
434     allDone.WaitOne(DefaultTimeout);
435     // reset handle
436     allDone.Reset();
437 }
438
439 public void getTasks(){
440     req = null;
441     req = (HttpWebRequest)WebRequest.Create(@"http://" +
442     serverIp +
443     ":" + serverPort + "/engine-rest/task");
444     req.Method = "GET";

```

```

431     req.Headers["Authorization"] = "Basic " + Convert.
432         ToBase64String(
433             Encoding.UTF8.GetBytes(User.Id.ToString() + ":" +
434                 User.Password.ToString())));
435
436     Tasks = new camundaTasks();
437     // Adding type of response what are we waiting for
438     processType pt = processType.taskList;
439     // Start the asynchronous request.
440     req.BeginGetResponse(new AsyncCallback(FinishWebRequest)
441         , Tuple.Create(req, pt));
442     //req.BeginGetResponse(new AsyncCallback(FinishWebRequest),
443     //    req);
444     // set default state
445     lastState = (int)returnStates.timeout;
446     // waiting for result
447     allDone.WaitOne(DefaultValue.Timeout);
448     // reset handle
449     allDone.Reset();
450 }
451
452 public void authenticate(){
453     req = null;
454     req = (HttpWebRequest)WebRequest.Create(@"http://" +
455         serverIp + ":" + serverPort + "/engine-rest/user/" +
456         User.Id + "/profile");
457     req.Method = "GET";
458     req.Headers["Authorization"] = "Basic " + Convert.
459         ToBase64String(Encoding.UTF8.GetBytes(User.Id + ":" +
460             User.Password)));
461
462     // Adding type of response what are we waiting for
463     processType pt = processType.userInfo;
464     // Start the asynchronous request.
465     req.BeginGetResponse(new AsyncCallback(FinishWebRequest)
466         , Tuple.Create(req,pt));
467     // set default state
468     lastState = (int)returnStates.timeout;
469     // wait for result
470     allDone.WaitOne(DefaultValue.Timeout);
471     // reset handle
472     allDone.Reset();
473 }
474
475 private async void FinishWebRequest(IAsyncResult result){
476     Tuple<HttpWebRequest, processType> state = (Tuple<
477         HttpWebRequest, processType>)result.AsyncState;
478     HttpWebResponse response = null;
479     try{
480         response = state.Item1.EndGetResponse(result) as
481             HttpWebResponse;
482
483         using (var reader = new StreamReader(response.
484             GetResponseStream(), Encoding.UTF8)){
485             string responseText = reader.ReadToEnd();
486             if (state.Item2 == processType.userInfo)
487                 parseUser(responseText);

```

```

477             else if (state.Item2 == processType.taskList)
478                 parseTaskList(responseText);
479             else if (state.Item2 == processType.variables)
480                 parseVariables(responseText);
481             else if (state.Item2 == processType.comment)
482                 parseComments(responseText);
483             else if (state.Item2 == processType.processes)
484                 parseProcList(responseText);
485             else if (state.Item2 == processType.
486                 processesVariables)
487                 parseProcVars(responseText);
488             else{
489                 lastState = (int) returnStates.unknownError;
490             }
491             lastState = (int) returnStates.allIsOk;
492         }
493     }
494     catch (WebException webEx){
495         if (webEx.Response != null){
496             response = (HttpWebResponse)webEx.Response;
497             if (response.StatusCode == HttpStatusCode.
498                 Unauthorized)
499                 lastState = (int) returnStates.unauthorized;
500             else
501                 lastState = (int) returnStates.unknownError;
502         }else{
503             lastState = (int) returnStates.unreachable;
504         }
505     }finally{
506         allDone.Set();
507     }
508 }
509 private void parseComments(string input)
510 {
511     if (input == null)
512         return;
513     Tasks.Data.ElementAt(lastElement).comments = new List<
514         comment>();
515     var objects = JArray.Parse(input);
516     foreach ( JObject jo in objects)
517     {
518         comment temp = JsonConvert.DeserializeObject<comment>(jo.ToString());
519         Tasks.Data.ElementAt(lastElement).comments.Add(temp)
520     }
521 }
522 private void parseVariables(string input)
523 {
524     if (input == null)
525         return;
526     if (input == "[]")
527         return;
528
529     Tasks.Data.ElementAt(lastElement).Variables = new List<

```

```

        variable>();

530    IDictionary<string, JToken> Jsondata = JObject.Parse(
531        input);
532    foreach (KeyValuePair<string, JToken> element in
533        Jsondata){
534        string innerKey = element.Key;
535        variable temp= JsonConvert.DeserializeObject<
536            variable>(element.Value.ToString());
537        temp.name = innerKey;
538        Tasks.Data.ElementAt(lastElement).Variables.Add(temp
539        );
540    }
541 }
542 private void parseTaskList(string input)
543 {
544     if (input == null)
545         return;
546     Tasks.Data = new List<camundaTask>();
547     var objects = JArray.Parse(input);
548     foreach (JObject jo in objects){
549         camundaTask temp = JsonConvert.DeserializeObject<
550             camundaTask>(jo.ToString());
551         Tasks.Data.Add(temp);
552     }
553 }
554 private void parseProcVars(string input)
555 {
556     if (input == null)
557         return;
558     if (input == "{}")
559         return;
560
561     procs.data.ElementAt(lastProcKeyId).variables =
562         new List<variable>();
563
564     IDictionary<string, JToken> Jsondata = JObject.
565         Parse(input);
566     foreach (KeyValuePair<string, JToken> element in
567         Jsondata)
568     {
569         string innerKey = element.Key;
570         variable temp = JsonConvert.
571             DeserializeObject<variable>(element.
572                 Value.ToString());
573         temp.name = innerKey;
574
575         procs.data.ElementAt(lastProcKeyId).
576             variables.Add(temp);
577     }
578 }
579 private void parseProcList(string input)
580 {
581     if (input == null)
582         return;
583 }
```

```

576     procs.data = new List<camundaProc>();
577     var objects = JArray.Parse(input);
578     List<string> added = new List<string>();
579     foreach ( JObject jo in objects)
580     {
581         camundaProc temp = JsonConvert.DeserializeObject<
582             camundaProc>(jo.ToString());
583         if (!added.Contains(temp.name))
584         {
585             added.Add(temp.name);
586             procs.data.Add(temp);
587         }
588     }
589     private void parseUser(string input)
590     {
591         if (String.IsNullOrEmpty(input))
592             return;
593
594         input = input.Substring(1, input.Length - 2);
595         string[] properties = input.Split(',');
596         for (int i = 0; i < properties.Length; i++)
597         {
598             string[] values = properties[i].Split(':');
599             values[0] = values[0].Substring(1, values[0].Length
600                 - 2);
601             values[1] = values[1].Substring(1, values[1].Length
602                 - 2);
603             if (values[0] == "firstName")
604                 user.FirstName = values[1];
605             else if (values[0] == "lastName")
606                 user.LastName = values[1];
607             else if (values[0] == "email")
608                 user.Email = values[1];
609         }
610     }
611 }

```

### Листинг Д.3. fileDownloader.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using Xamarin.Forms;
7
8  namespace camundaClient
9  {
10     public interface ISave
11     {
12         void LoadFile(string filename, Uri url, string user, string
13             password);
14         int getProgress();
15         bool isFinished();
16         bool isSuccessed();
17         void cancel();
18     }
19 }

```

```

17     }
18
19     public class fileDownloader
20     {
21         private string serverPort { get; set; }
22         private string serverIp { get; set; }
23         private string userId { get; set; }
24         private string userPassword { get; set; }
25
26         public fileDownloader(string serverPort, string serverIp,
27                               string userId, string userPassword)
28         {
29             this.serverPort = serverPort;
30             this.serverIp = serverIp;
31             this.userId = userId;
32             this.userPassword = userPassword;
33         }
34
35         public ISave ISS = DependencyService.Get<ISave>();
36
37         public async void getFile(string typeName, string fileName,
38                                   string id, bool isCase)
39         {
40             Uri url;
41             if (isCase)
42                 url = new Uri("http://" + serverIp +
43                             ":" + serverPort + "/engine-rest/case-instance/" +
44                             id +
45                             "/variables/" + typeName + "/data");
46             else
47                 url = new Uri("http://" + serverIp +
48                             ":" + serverPort + "/engine-rest/process-instance/" +
49                             id +
50                             "/variables/" + typeName + "/data");
51             await Task.Run(() => ISS.LoadFile(fileName, url, userId,
52                                                 userPassword));
53         }
54     }
55 }
```

Листинг Д.4. MainPage.xaml.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Linq;
5  using System.Runtime.CompilerServices;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Windows.Input;
9  using Xamarin.Forms;
10
11 namespace camundaClient
12 {
13     public partial class MainPage : ContentPage
14     {
```

```

15     private camundaUser userData;
16     private engine engine;
17
18     public MainPage(){
19         InitializeComponent();
20         //serverIp.Text = "192.168.0.103";
21         //serverPort.Text = "8080";
22         //userId.Text = "demo";
23         //userPassword.Text = "demo ";
24         signin.Clicked += Signin_Clicked;
25     }
26
27     private string warning = "Warning";
28
29     private void Signin_Clicked(object sender, EventArgs e){
30         if (serverIp.Text.Equals(String.Empty)){
31             notify(warning, "Server ip is empty");
32             return;
33         }
34         if (serverPort.Text.Equals(String.Empty)){
35             notify(warning, "Server port is empty");
36             return;
37         }
38         if (userId.Text.Equals(String.Empty)){
39             notify(warning, "User id is empty");
40             return;
41         }
42         if (userPassword.Text.Equals(String.Empty)){
43             notify(warning, "User password is empty");
44             return;
45         }
46
47         if (userData == null)
48             userData = new camundaUser(userId.Text.ToString(),
49                                         userPassword.Text.ToString());
50         else{
51             userData.Id = userId.Text;
52             userData.Password = userPassword.Text;
53         }
54         if (engine == null)
55             engine = new engine(serverIp.Text, serverPort.Text,
56                                  userData);
57         else{
58             engine.serverIp=serverIp.Text;
59             engine.serverPort=serverPort.Text;
60             engine.User = userData;
61         }
62
63         overlayOn();
64         auth();
65     }
66     private void overlayOn(){
67         inputForm.Opacity = 0.3;
68     }
69     private void overlayOff(){
70         inputForm.Opacity = 1;
71     }

```

```

71     private async Task auth()
72     {
73         await Task.Run(()=>engine.authenticate());
74         int result = engine.lastState;
75         if (result == 0){
76             loadTasks();
77             return;
78         }
79         else if (result == 1)
80             notify(warning, "Request timeout explained.\n Try
81                 again later!");
82         else if (result == 2)
83             notify(warning, "Wrong user id or password!");
84         else if (result == 3)
85             notify(warning, "Server unreachable!");
86         else
87             notify(warning, "Unknown error!");
88
89         overlayOff();
90     }
91     private async Task loadTasks(){
92         await Task.Run(() => engine.getTasks());
93
94         int result = engine.lastState;
95         if (result == 0){
96             overlayOff();
97             Navigation.PushModalAsync(new workspace(engine));
98             return;
99         }
100        else if (result == 1)
101            notify(warning, "Request timeout explained.\n Try
102                 again later!");
103        else if (result == 2)
104            notify(warning, "Wrong user id or password!");
105        else if (result == 3)
106            notify(warning, "Server unreachable!");
107        else
108            notify(warning, "Unknown error!");
109
110        overlayOff();
111    }
112    private void notify(string header, string msg){
113        DisplayAlert(header, msg, "Ok");
114    }
115 }
```

Листинг Д.5. ValueDataConverter.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using Newtonsoft.Json;
7  using Newtonsoft.Json.Linq;
```

```

8
9 namespace camundaClient
10 {
11     public class ValueDataConverter : JsonConverter
12     {
13         public override bool CanWrite { get { return false; } }
14
15         public override bool CanConvert(Type objectType)
16         {
17             return (objectType == typeof(variable));
18         }
19         public override object ReadJson(JsonReader reader, Type
20             objectType, object existingValue, JsonSerializer
21             serializer)
22         {
23             JObject item = JObject.Load(reader);
24             variable temp = new variable();
25             temp.type = item["type"].ToObject<string>();
26             temp.valueInfo = JsonConvert.DeserializeObject<valueInfo
27                 >(item["valueInfo"].ToString());
28             if (item["value"].Type == JTokenType.String)
29             {
29                 temp.value = item["value"].ToObject<string>();
30                 temp.valType = valueType.simpleString;
31             }
32             else if (item["value"].Type == JTokenType.Float)
33             {
34                 temp.floatValue = item["value"].ToObject<float>();
35                 temp.valType = valueType.d_number;
36             }
37             else if (item["value"].Type == JTokenType.Array)
38             {
39                 temp.values = item["value"].ToObject<List<string>>()
40                     ;
41                 temp.valType = valueType.listString;
42             }
43             return temp;
44         }
45
46         public override void WriteJson(JsonWriter writer, object
47             value, JsonSerializer serializer)
48         {
49             throw new NotImplementedException();
50         }
51     }
52 }

```

Листинг Д.6. UITests.cs

```

1 using System;
2 using System.IO;
3 using System.Linq;
4 using NUnit.Framework;
5 using Xamarin.UITest;
6 using Xamarin.UITest.Queries;
7
8 namespace Camunda.UITests

```

```

9   {
10    [TestFixture(Platform.Android)]
11    ///[TestFixture(Platform.iOS)]
12    public class Tests
13    {
14
15      // Server information
16      private string trustedIp = "192.168.0.107";
17      private string trustedPort = "8080";
18
19      // User credentials
20      private string trustedUserId = "demo";
21      private string trustedUserPassword = "demo";
22      private string userFirstName = "First name";
23      private string userLastName = "Last name";
24      private string userEmail = "demo@demo.com";
25
26      // For new task
27      private string newTaskName = "testTaskName";
28      private string newTaskDesc = "testTaskDesc";
29      private string newTaskComment = "testComment";
30
31      IApp app;
32      Platform platform;
33
34      public Tests(Platform platform)
35      {
36          this.platform = platform;
37      }
38
39      [SetUp]
40      public void BeforeEachTest()
41      {
42          //app=ConfigureApp.Android.StartApp();
43          app = AppInitializer.StartApp(platform);
44      }
45
46      [Test]
47      public void AppLaunches()
48      {
49          app.Screenshot("First screen.");
50      }
51
52      [Test]
53      public void authenticate()
54      {
55          app.EnterText("autoServerIp", trustedIp);
56          app.EnterText("autoServerPort", trustedPort);
57          app.EnterText("autoUserId", trustedUserId);
58          app.EnterText("autoUserPassword", trustedUserPassword);
59
60          app.Tap("autoSignIn");
61
62          app.WaitForElement(x => x.Marked("autoUserCred"),
63                             timeout: TimeSpan.FromSeconds(10));
64          app.Repl();
65      }
66      [Test]

```

```

66     public void createTestTask()
67     {
68         app.EnterText("autoServerIp", trustedIp);
69         app.EnterText("autoServerPort", trustedPort);
70         app.EnterText("autoUserId", trustedUserId);
71         app.EnterText("autoUserPassword", trustedUserPassword);
72
73         app.Tap("autoSignIn");
74
75         app.WaitForElement(x => x.Marked("autoUserCred"),
76                             timeout: TimeSpan.FromSeconds(10));
77
78         app.Tap("autoCreateTask");
79
80         app.EnterText("autoNewTaskName", newTaskName);
81         app.EnterText("autoNewTaskDecr", newTaskDesc);
82
83         app.Tap("autoBtnCreateTask");
84
85         app.WaitForElement("Task created successfully", "Not
86                             found", TimeSpan.FromSeconds(10));
87     }
88
89     [Test]
90     public void claimTestTask()
91     {
92         app.EnterText("autoServerIp", trustedIp);
93         app.EnterText("autoServerPort", trustedPort);
94         app.EnterText("autoUserId", trustedUserId);
95         app.EnterText("autoUserPassword", trustedUserPassword);
96
97         app.Tap("autoSignIn");
98
99         app.WaitForElement(x => x.Marked("autoUserCred"),
100                            timeout: TimeSpan.FromSeconds(10));
101
102         app.Tap(x => x.Marked("All Tasks"));
103         app.Tap(x => x.Class("ListView").Child().Index(0));
104
105        app.WaitForElement(newTaskName, "Not found", TimeSpan.
106                           FromSeconds(10));
107        app.Tap("autoBtnClaimUnclaim");
108        app.WaitForElement("Task claimed successfully", "Not
109                             found", TimeSpan.FromSeconds(10));
110    }
111    [Test]
112    public void addComment()
113    {
114        app.EnterText("autoServerIp", trustedIp);
115        app.EnterText("autoServerPort", trustedPort);
116        app.EnterText("autoUserId", trustedUserId);
117        app.EnterText("autoUserPassword", trustedUserPassword);
118
119        app.Tap("autoSignIn");
120
121        app.WaitForElement(x => x.Marked("autoUserCred"),
122                            timeout: TimeSpan.FromSeconds(10));

```

```

118     app.Tap(x => x.Marked("All Tasks"));
119     app.Tap(x => x.Class("ListView").Child().Index(0));
120
121     app.WaitForElement(newTaskName, "Not found", TimeSpan.
122         FromSeconds(10));
123
124     app.Tap(x => x.Marked("Comments"));
125
126     app.EnterText("autoLblCommentText", newTaskComment);
127     app.Tap("autoBtnAddComment");
128
129     app.WaitForElement("Comment added successfully", "Not
130         found", TimeSpan.FromSeconds(10));
131 }
132
133 [Test]
134 public void completeTask()
135 {
136     app.EnterText("autoServerIp", trustedIp);
137     app.EnterText("autoServerPort", trustedPort);
138     app.EnterText("autoUserId", trustedUserId);
139     app.EnterText("autoUserPassword", trustedUserPassword);
140
141     app.Tap("autoSignIn");
142
143     app.WaitForElement(x => x.Marked("autoUserCred"),
144         timeout: TimeSpan.FromSeconds(10));
145
146     app.Tap(x => x.Marked("My Tasks"));
147     app.Tap(x => x.Class("ListView").Child().Index(0));
148
149     app.WaitForElement(newTaskName, "Not found", TimeSpan.
150         FromSeconds(10));
151     app.Tap("autoBtnComleteTask");
152     app.WaitForElement("Task completed successfully", "Not
153         found", TimeSpan.FromSeconds(10));
154 }
155 }
```