ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(НИУ «БелГУ»)

ИНСТИУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ОБЩЕЙ МАТЕМАТИКИ

НОТИФИКАЦИОННЫЙ ФУНКЦИОНАЛ МЕДИЦИНСКОГО ПРИЛОЖЕНИЯ «MEDINFORM»

Выпускная квалификационная работа
обучающегося по направлению подготовки
01.03.02 Прикладная математика и информатика
очной формы обучения, группы 07001406
Кристалова Дмитрия Олеговича

Научный руководитель
д.т.н., профессор
Аверин Г.В.

БЕЛГОРОД 2018

Содержание

ВВЕДЕНИЕ

Нынешняя информационная эпоха постоянно влияет не только на медицину. В современном мире человеку недостаточно получения консультации и последующей выписки рецепта по болезни. Человек в современном обществе точно хочет знать, кто, как и от чего его лечит. Он желает получать консультации высококвалифицированных специалистов, покупать препараты исключительно проверенных производителей, иметь доступ к адаптированной под современные реалии медицинской литературе и быть в курсе новостей, касающихся медицинской сферы.

Однако, все перечисленные выше пункты хоть и важны, но не являются основополагающими. Темп жизни современного общества непрерывно увеличивается, что ведет к тому, что человеку требуются инструменты, которые позволят ему тратить минимальное количество времени на решение повседневных вопросов. Это приводит к необходимости создания такого функционала, как, к примеру система электронных очередей, приобретающая все большее распространение в российских медицинских учреждениях, а также к насыщению рынка приложений медицинской тематики, в первую очередь для смартфонов, как для электронных устройств, постоянно находящихся под рукой и имеющих доступ к сети интернет.

Работники здравоохранения, также хотят получать лёгкий и быстрый доступ к тематическим ресурсам и иметь возможность виртуального обращения к пациенту. Это позволяет сократить затраты времени на решение повседневных обязанностей.

Исходя из сказанного выше целью работы является создание блока рассылки уведомлений пользователям в разрезе медицинского приложения.

Основные задачи, решаемые в работе:

- изучить библиографические источники по теме ВКР;

- рассмотреть и изучить рынок приложений медицинской направленности и методы уведомления пользователя, используемые в подобных приложениях;
- исследовать возможные платформы разработки;
- изучить концепции о подходе к разработке;
- используя выбранную платформу запрограммировать технический блок;
- обеспечить взаимодействие блока со внешними сервисами;
- провести тестирование блока.

Объектом дипломного исследования являются нотификационные механизмы в приложениях медицинской направленности. Предмет исследования – реализация программного блока способного качественно выполнять задачу уведомления пользователей.

Структура и объем работы: выпускная квалификационная работа выполнена на 70 страницах машинописного текста. Состоит из введения, четырех разделов, заключения и приложений.

В первом разделе изучаются существующие программные продукты администрирования приема препаратов, рассматриваются методы решения задачи уведомления пользователя.

Во втором разделе изучается концепция BPM, проводится сравнительный анализ различных BPM платформ, разрабатываются концептуальные программно-технические решения приложения.

В третьем разделе приводится описание практической части работы, внутренняя архитектура программного блока, сравнительный анализ сервисов рассылки смс сообщений.

В четвертом разделе реализуется тестирование программного блока.

В заключении по итогам проделанной работы, сформулированы выводы.

В приложениях представлены справочные таблицы, схемы, экранные формы, элементы кода.

РАЗДЕЛ 1. СОСТОЯНИЕ ВОПРОСА

1.1 Обзор программных продуктов администрирования приема препаратов

Порядка 20-25% владельцев смартфонов используют минимум одно мобильное приложение, которое можно охарактеризовать как медицинское. По различным оценкам, основанным на количестве скачиваний приложений с тегом «медицина» с iTunes или Google Play, в 2015 году количество пользователей таких приложений по миру уже достигало минимум 500 миллионов. Данное обстоятельство говорит о том, что популярность подобных приложений неуклонно растет, однако приносят ли они фактическую пользу нашему здоровью? Возможно ли, что в действительности эффективность и полезность подобных приложений крайне мала?

По терминологии «Управления по контролю за продуктами и лекарствами США» (FDA), мобильное медицинское приложение - «устройство медицинского назначения, являющееся мобильным приложением, соответствующее определению устройства медицинского назначения и являющиеся дополнением к сертифицированному устройству медицинского назначения или трансформирующие платформу мобильного телефона в сертифицированное устройство медицинского назначения».

На сегодняшний день рынок медицинских приложений огромен, он насчитывает десятки тысяч приложений различной направленности: позволяющие осуществлять мониторинг практически каждого параметра состояния здоровья массы тела, физической нагрузки, артериального давления, уровней холестерина и глюкозы в крови, частоты сердечных сокращений и качества сна, различные сборники информации медицинской направленности, аптечки, системы уведомлений и многое другое. Данный список продолжает непрерывно увеличиваться.

Таким образом пользователи смартфонов могут получать значительное количество информации о состоянии своего здоровья и следить за ним, при этом реже обращаясь к специалистам. Однако по мнению специалистов эффективность подобных приложений вызывает определенные сомнения.

Можно без доли скептицизма заявлять, что не все приложения одинаково полезны. Больше половины существующих приложений следует отнести к развлекательным, а не образовательным, а тем более действительно медицинским. Институт мединформации (IMS) провел анализ данных приложений основываясь на выборке из сорока тысяч, доступных для скачивания на iTunes и Google Play, в результате было выявлено, что порядка 75-80% данных приложений относятся к развлекательным, еще 15-20% относятся к информационным и лишь оставшиеся 5% можно так или иначе отнести в группу медицинских.

Несмотря на то, что данная выборка проводилась не только в интересующем нас подразделе медицинских приложений, она очень хорошо иллюстрирует общее состояние рынка. Несмотря на кажущуюся наполненность рынка, фактически действительно полезных приложений единицы, а следовательно, качественное приложение в данной отрасли будет пользоваться спросом.

Оценив российский сегмент данного рынка определяем, что существует всего 2 приложения имеющих схожий с разрабатываемым в работе приложением, функционал, минимально удобный интерфейс, а также поддерживаемые производителями хотя-бы в минимальном объеме.

Первое приложение – *Ваш провизор – «аптека и сканер»*. Данное приложение было разработано в 2016 году, последнее обновление производилось в октябре 2017 года. Данное приложение имеет следующий функционал:

- проверка подлинности лекарства определяется при помощи штрих-кода на упаковке;

- ведение учета лекарств в домашней аптечке (название, тип, количество, срок годности);
- контроль времени приема препаратов пользователем при помощи внутреннего календаря;
- база общей информации о препаратах: инструкция, противопоказания и дозировки;
- встроенная опция нахождения ближайших аптек.

Функционал данного приложения достаточно широк, а внутренний интерфейс удобен в использовании, однако существует определенный перечень недостатков, которые значительно влияют на общее качество приложения: проверка подлинности и ведение учета доступно исключительно для препаратов, находящихся во внутренней базе информации о препаратах, которая обновляется медленно и несвоевременно. Опция нахождения ближайших аптек работает значительно хуже, чем аналогичные сервисы 2ГИС, Google и Яндекс карт, видимо используя некие внутренние решения, а потому качество работы данного функционал гарантируется исключительно в пределах Москвы и Санкт-Петербурга. Касательно пункта об информировании пользователя о приеме препарата, представляющего наибольший интерес в разрезе данной работы, стоит заметить, что время приема препарата имеет достаточно грубую временную настройку, дата начала приема работает несколько иначе чем ожидает пользователь, а функционал «уведомлений» ограничен занесением точек приема во внутренний календарь приложения. Также отсутствует обратная связь между пользователями и сопровождающими данный программный продукт лицами.

Второе приложение – «Моя аптечка». Несмотря на название данное приложение представляет следующий функционал: поиск препарата по базе данных, предоставление информации о препарате, система уведомлений. В данном приложении отсутствуют элементы функционала, отвечающие за

отслеживание назначенных приемов, а уведомление ведется при помощи Push сообщений.

1.2 Существующие методы разрешения задачи уведомления пользователя

Стоит выделить набор наиболее часто применяемых методов уведомления пользователя о произошедшем событии:

- Email сообщение;
- SMS сообщение;
- голосовой вызов;
- RSS;
- Push сообщение.

В данной работе не будет рассматриваться метод уведомления при помощи голосового вызова, так как для своего использования он требует наличия сотрудника(-ов) для реализации, а потому избыточен в рамках создаваемого приложения на данном этапе разработки. Далее мы рассмотрим остальные методы.

Электронная почта (*email*) – служба и технология пересылки и получения электронных сообщений между различными пользователями компьютерной сети.

Электронная почта по составу своих элементов, а также принципу работы практически полностью повторяет систему обычной почты, заимствуя термины и характерные особенности - простота использования, множественные задержки передачи сообщений, достаточная надёжность и в то же время отсутствие гарантии доставки.

Электронная почта имеет следующие достоинства: достаточно легко воспринимаемые, а также запоминаемые человеком адреса конкретного вида, возможность передачи простого и форматированного текста, а также произвольных файлов, независимость серверов, высокая надёжность доставки сообщения, простота использования, высокая скорость передачи сообщений.

Электронная почта имеет следующие недостатки: наличие спама, в определенной степени возможные задержки доставки сообщения, а также ограничения на размер сообщения и на общий размер всех сообщений в почтовом ящике.

Данный вариант рассылки уведомлений будет использоваться в приложении несмотря на то, что в первом приближении он не выглядит оптимальном для рассылки сообщений подобного типа, по следующим причинам: теоретически возможна ситуация отсутствия у пользователя доступа к телефону, но наличия компьютера с доступом к сети интернет, а также для дублирования отправляемого сообщения. Также наличие подобного функционала может оказаться важным или даже необходимым в случае расширения функционала приложения, а потому наличие настроенного канала уведомления посредством электронной почты оправдано.

СМС - технология, позволяющая осуществлять приём и передачу коротких текстовых сообщений с помощью мобильного телефона. Входит в стандарты сотовой связи.

Технология смс имеет следующие определяющие характеристики:

- SMS, доставляются в течение не более чем 10 секунд. Отправитель имеет возможность получать уведомление о доставке сообщения;
- сообщение можно отправить на находящийся вне зоны действия сети или выключенный телефон. Как только адресат появляется в зоне действия сети, он получает сообщение. Получение отправителем уведомления о доставке, зафиксирует момент появления получателя в сети;
- сообщение может быть отправлено абоненту, в данный момент занятому разговором;
- особенность технологии такова, что передача SMS почти никак не нагружает сотовую сеть.

Технология SMS поддерживается основными сотовыми сетями (GSM, NMT, D-AMPS, CDMA, UMTS).

Также SMS на телефоны можно отправлять из интернета и из других сетей используя специальные программы, универсальные SMS-формы, а также непосредственно смс шлюзы мобильных операторов.

Данный канал отправки уведомлений будет использоваться в качестве основного, подробнее об этом будет написано в разделе 3.3.

RSS — это группа XML-форматов, использующихся для полного или краткого изложения анонсов, новостей, лент статей, новых постов и прочем.

Аббревиатура RSS расшифровывается по-разному, в зависимости от версии:

- RSS 0.9x — обогащенная сводка новостей сайта;
- RSS 0.9 и 1.0 — сводка ленты записей сайта с использованием мета структуры описания вебсайтов;
- RSS 2.x — реально простое распространение.

Прямым предназначением RSS 2.0 является - транслировать и публиковать краткое описание информации, а также ссылку на полную версию контента. Тем не менее, стоит упомянуть, что часто сайты отдают статьи полностью напрямую в RSS-ленту, для обеспечения удобства пользователей. От этого теряется определенная доля посещаемости и дохода, но обеспечивается повышение аудитории за счет большего удобства. Другие сайты действуют исходя из обратного - анонс новой статьи заканчивается на самом интересном моменте, вынуждая читателя переходить по ссылке для просмотра полной версии контента, однако данный подход негативно воспринимается пользователями.

В данной работе не будут перечислены браузеры, которые умеют работать с RSS-лентами, потому что в текущий момент времени это реализуют все браузеры. Загружать данные из RSS могут и почтовые клиенты такие как Thunderbird или Outlook.

Данная же система не будет применяться в связи с несоответствием специфике разрабатываемого приложения.

Технология Push - один из способов распространения контента в сети интернет, при котором данные поступают от поставщика к пользователю на основе четко установленных параметров. Пользователь, в свою очередь, имеет возможность принять или отвергнуть информацию.

Среднестатистический пользователь может подписываться на различные темы, получать информацию от сервиса провайдера, и постоянно, когда каждое новое обновление формируется на сервере, обновление доставляется на пользовательский компьютер. Противоположностью же технологии Push сообщений является Pull технология, в которой запрос инициирует клиентское ПО.

Push технологии приобрели столь широкую известность благодаря продукту PointCast, популярному в 1990-е годы. Сеть PointCast занималась следующим: доставка новостей, доставка данных фондовых рынков, содержание агрегатора с собственным форматом, напоминавшим телевидение, с рисунками и текстами, вместо видео. Влияние СМИ было крайне значительным, вследствие чего корпорации Netscape и Microsoft в разгар браузерного противостояния решили включить эту технологию в свои браузеры. Однако стоит заметить, что в большинстве случаев пользователи имели низкую скорость подключения, поэтому соответственно и популярность сервиса была крайне низкой, а позже и вовсе сошла на нет, вытесненная pull-технологией RSS в начале 2000-х годов.

Для работы push-уведомлений используются следующие компоненты:

- сервер push-уведомлений;
- сервер автора приложения, посылающий уведомления серверу push-уведомлений;
- непрерывно работающая служба в операционной системе устройства, общающаяся с сервером push-уведомлений;
- приложение, поддерживающее push-уведомления.

Разработчик приложения предварительно регистрирует свой сервер на сервере уведомлений ОС.

После того как пользователем было дано разрешение приложению на получение уведомлений, данное приложение отправляет уникальный ID и уникальный номер устройства на сервер уведомлений, а также регистрируется на представленном сервере. В последствии эти два уникальных номера образуют уникальный идентификатор. Позднее данный идентификатор будет отправляться с сервера уведомлений на сервер владельца приложения.

В тот момент, когда сервер создателя приложения должен будет отправить уведомление кому-либо из клиентов, он сформирует непосредственно сообщение и список уникальных идентификаторов, а затем отправит эти данные с помощью специализированного API на сервер уведомлений. Сервер пересылает эти сообщения клиентуре. Клиенты вправе как отклонить, так и принять данные.

Уведомления могут содержать различные поля: кнопки ответа, изображения, числовые значения, звук и прочее.

Самое известное использование Push - рассылка подписочных сообщений, информационных бюллетеней, доставляемых по электронной почте. Подобная система используется в судебных учреждениях США, которые отправляют на электронную почту подписчиков информацию о процессах.

Типичные примеры push-сервисов:

- синхронные конференции и системы обмены мгновенных сообщений;
- система умформеров (автоматическим образом обновляющийся блок контента со стороны поставщика, устанавливаемый на сайте пользователя);
- SMTP-системы электронную почту также относятся к push-системам.

Push запросы могут быть смоделированы также и с помощью регулярных запросов pull типа, как это происходит, при извлечении сообщений электронной почты с сервера POP3, когда почтовый клиент делает запросы ежеминутно.

Следует упомянуть, такие системы, как Kazza, которая включает в себя push-технологию файлов дольщиков, где присутствует возможность выбрать любой контент-сервер, который впоследствии будет подключён.

Другие виды использования push-технологии включают в себя веб-приложения, в том числе распространения данных рынка, мониторинг сетевых данных, аукционы и прочее.

Также наличествуют специальные сайты, позволяющие автоматизировать процесс отправки push-уведомлений.

1.3 Выводы и задачи работы.

Таким образом можно заключить, что в объективной реальности существует потребность в медицинских приложениях, доступных для использования непрофессионалами данной сферы. Также стоит обратить внимание на то, что потребность в подобных приложениях непрерывно растет. Можно заключить, что существует как минимум две объективных причины расширения рынка медицинских приложений.

Во-первых, увеличение количества людей, пользующихся смартфонами. Данный аспект ведет за собой также расширение доступной пользовательской аудитории, так как наличествует не только расширение пользователей в конкретных возрастных сегментах, но и увеличение количества этих сегментов, с течением времени нижний и верхний возрастные пороги постоянно расширяются.

Во-вторых, увеличивается заинтересованность людей в сопровождении собственного здоровья, а также их информированность в данном вопросе. Является это позитивной или негативной тенденцией вполне может стать темой отдельной исследовательской работы.

Данные факторы определяют первую причину актуальности данной работы. Однако существуют также и прочие причины. Одна из них – фактическое наличие рынка сбыта данных приложений, спрос рождает предложение и если мы видим рост спроса, то темпы роста предложения значительно ниже, а качество предполагаемой продукции зачастую недостаточно. А также данная тема актуальна в разрезе применяемых платформ и подходов, о чем будет сказано далее. Основная платформа разработки достаточно широко применяется за рубежом и только начинает проникать в российское ИТ, а следовательно, любой опыт работы с ней может оказаться крайне востребованным.

Также в ходе выполнения и подготовки данного диплома предоставляется возможность ознакомиться со специфическими, но при этом достаточно распространенными задачами реальной практики. Ознакомиться с различными сервисами уведомлений, определить сферы их эффективного применения, плюсы и минусы каждого направления.

Исходя из сказанного выше сформулируем следующие задачи работы:

- изучить библиографические источники по теме ВКР;
- рассмотреть и изучить рынок приложений медицинской направленности и методы уведомления пользователя, используемые в подобных приложениях;
- исследовать возможные платформы разработки;
- изучить концепции о подходе к разработке;
- используя выбранную платформу запрограммировать технический блок;
- обеспечить его взаимодействие со внешними сервисами;
- провести тестирование блока.

РАЗДЕЛ 2. ОПРЕДЕЛЕНИЕ ПЛАТФОРМЫ РАЗРАБОТКИ

2.1 Концепция bpm (business process management)

Концепция BPM была создана, включая в себя BI-системы и приложения аналитики. В определенный момент времени появилась существенная необходимость интеграции, методологической и технологической. Новое направление получило название Business Performance Management (BPM), что переводится как "Управление Эффективностью Бизнеса". BPM – «целостный, процессно-ориентированный подход к принятию управленческих решений, направленный на улучшение способности компании оценивать свое состояние и управлять эффективностью своей деятельности на всех уровнях, путем объединения собственников, менеджеров, персонала и внешних контрагентов в рамках общей интегрированной среды управления».[5]

В современном мире принято несколько другое определение BPM.

BPM - «методология, направленная на оптимизацию реализации стратегии и состоящая из набора интегрированных циклических аналитических процессов, которые поддерживаются соответствующими технологиями и имеют отношение как к финансовой, так и к операционной информации. BPM позволяет предприятию определять, измерять и управлять эффективностью своей деятельности, направленной на достижение стратегических целей. Ключевые финансовые и операционные процессы BPM включают планирование, консолидацию и отчетность, анализ ключевых показателей эффективности и их распространение в рамках организации».[2]

Понятие BPM может сразу в нескольких значениях: во-первых – управленческая концепция, во-вторых - комплекс программно-технических средств, обеспечивающих поддержку идеологии BPM и реализующих ее на практике.

Однако все это привело к тому, что разные организации могут применять различные определения для обозначения одной и той же

концепции. Сегодня можно встретить, не менее чем, четыре различные аббревиатуры:

- управление эффективностью бизнеса (Business Performance Management, BPM);
- управление эффективностью деятельности предприятия (Enterprise Performance Management, EPM);
- управление эффективностью деятельности корпорации (Corporate Performance Management, CPM);
- стратегическое управление предприятием (Strategic Enterprise Management, SEM).

Следует отметить следующий нюанс: BPM, как аббревиатура, имеет второй вариант расшифровки - Business Process Management (управление бизнес-процессами). Этот термин используется, в частности разработчиками платформы Pega, на которой было реализовано приложение используемое в данной работе.[2]

Несмотря на терминологические нестыковки, определение BPM уже достаточно давно признано специалистами области управления и крупными компаниями. Концепция BPM превратилась в отдельное направление, имеющее собственную теоретическую идею, а также методики и технологии для ее реализации.

В соответствии с документом, разработанным Группой по стандартизации BPM, в качестве основных процессов, охватываемых BPM-системами, можно выделить следующие:

- формализация стратегии (strategize);
- планирование (plan);
- мониторинг и анализ (monitor and analyze);
- корректирующие воздействия (take corrective actions).

Стратегии BPM-систем предоставляют менеджерам право разрабатывать отдельные стратегии и распределять их на подразделения

фирмы, определять возможности создания стоимости и сформировывать четкие системы метрик, которые позволят оценить динамическую эффективность бизнеса.

В сфере планирования BPM-системы предоставляют менеджерам возможность устанавливать свои локализованные цели, разрабатывать программы, разрабатывать бюджеты, моделировать качественные сценарии планирования, поддерживающие стратегию бизнеса, формировать целевые значения локализованных показателей различных временных периодов.

Для мониторинга и анализа BPM-системы позволяют следующее: оценивать индивидуальную эффективность и групповую эффективность с применением определенных ключевых показателей на всех организационных уровнях.

Также BPM-системы помогают своим менеджерам вовремя реагировать на возникающие отклонения и прочие нестандартные ситуации.

Данная классификация представлена в соответствии со стандартным циклом стратегического управления: первые две группы процессов – целеполагание и трансформация стратегий в планы, вторые две группы – контроль, корректировка целей и планов. Классификация отражает структуру функциональных областей BPM. Однако данная классификация не подходит для структурирования информационных систем, обеспечивающих данные функции. Конкретные программные продукты реализуют более чем одну ключевую функцию.

Таким образом, концепция управления эффективностью бизнеса может применяться для предприятий и организаций самых разных отраслей, включая организации социальной сферы. Эта концепция имеет непосредственное отношение к стратегическому менеджменту, поскольку она предусматривает целый ряд важных управленческих функций, включая формализацию стратегии и определение ключевых показателей, планирование, мониторинг и анализ, а также обеспечение необходимой обратной связи и корректирующие воздействия. С другой стороны,

концепция BPM тесно связана с задачами корпоративного управления, позволяя обеспечить информационную прозрачность организации для заинтересованных лиц, в частности, путем формирования и представления корпоративной отчетности.

2.2 Обзор архитектуры pega и сравнение с прочими платформами разработки (специфика)

Понятие Business Process Management (BPM) с течением времени все активнее упоминается в среде различной величины корпораций. Ключевая идея данного понятия в том, что отдельные бизнес-процессы компании рассматриваются, как, доступные для использования активы, а также как объекты, которые могут увеличить прибыльность отдельно взятого бизнеса. Инструментарий, который будет использоваться для решения данных задач, может быть абсолютно любым, на выбор сотрудников: бумажный лист, текстовые документы, Visio или другое, подобное данному, программное обеспечение для создания схем или диаграмм и т.д. Однако существует инструментарий, который изначально создавался для того, чтобы использоваться в качестве инструмента решения данных задач и трансформировать бизнес — это BPM-платформы. [1]

Задача для BPM-платформы определяется двусторонне: сначала требуется визуализировать определенный бизнес-процесс, однако с другой стороны — существует необходимость исполнения данного процесса.

Рисунок 1.1 Gartner BPM Magic Quadrant 2014

Полностью описан весь рынок BPM-платформ, в статье независимого агентства Gartner [1], поэтому проведем краткое сравнение и описание основных моментов.

Кейс-менеджмент, основанный на BPM-Платформах призван помочь реализовать такие решения архитектуры, которые позволят создать систему уникальных и гибких решений для управления делами. На данный момент существует 11 основных поставщиков в этом растущем сегменте рынка.

В первую очередь следует определить каким образом будет проводится

сравнение:

BPM - это в первую очередь подход к сопровождению проекта, определенный образ мышления, а не исключительно платформа для автоматизации согласованного-написанного ТЗ. Как Pega, так и IBM в общем рекомендуют схожие подходы:

- постановка целей;
- итеративная разработка;
- легкость изменений.

Однако в данной работе будет обращено внимание на различные аспекты: процедуры управления проектом, требованиями, ожиданиями.

При всей кажущейся схожести платформ Pega и IBM, будет некорректно проводить сравнение исключительно базовых функциональных блоков данных платформ. Pega имеет множество дополнительных фреймворков, устанавливаемых отдельно. У IBM также имеются различные уровни лицензирования, а также другие платформы, дополняющие IBM BPM.

В данной работе обзоры Pega и IBM приводятся в несколько более широком формате, чем исключительно сравнение, BPM платформы - это комплексы методологических и программных средств:

- непосредственно краткие обзоры внутреннего строения Pega и IBM BPM, а также укороченное описание методологий их внедрения. Данные обзоры не могут претендовать на абсолютную полноту описания, а приводятся в справочных целях, но, они будут в немалой степени полезны для первичного погружения. Для получения более подробной информации необходимо обратиться к документации соответствующей платформы;
- основные сравнительные тезисы, сгруппированные в таблицу, вынесены в приложение А. Каждая строка представленной таблицы соответствует одному из сравниваемых аспектов или сравниваемых особенностей, а колонки соответственно соответствуют реализации данных аспектов непосредственно на разных платформах: Pega и IBM;

- уникальные преимущества для обоих платформ.

Для начала рассмотрим архитектуру Pega. Ядро Pega представляет из себя java-enterprise приложение, которое может быть запущено на любом application сервере. На базе данного ядра построен стек из классов, составляющих собой базовый фреймворк PRPC, включая как непосредственно сам портал разработчика, доступный пользователям через браузер (так как данный портал используется также непосредственно внутренней командой разработки, то в качестве вывода можно сказать, что Pega разработана на Pega).

Для понимания основных механизмов работы Pega, необходимо определить два базовых, в данной среде, понятия: класс и правило:

- класс - стандартная единица объектно-ориентированной парадигмы, представляющая из себя структуру, содержащую определенные связанные данные и методы их обработки;
- правило - экземпляр одного из специальных встроенных, определенных в одном из фреймворков, классов, назначаемый или присваиваемый к другим классам. Правила могут быть рассмотрены как реализация паттерна Стратегия, который так же может быть упомянут, как Behavior. Правило, принадлежащее определенному классу, определяет его свойство (property), поведение (flow action), способ отображения данных (section) и прочее.

Каждое Pega приложение состоит из набора некоторых классов, первая часть которых определяет структуру данных, эти классы хранятся в ветке Data-, вторая часть определяет структуру работ или кейсов и хранится в ветке Work-. Каждый из этих классов имеет возможность наследования свойств и методов, определяемых правилами, от классов, заданных во фреймворках, либо же от прочих классов приложения. Следует уточнить, что любой класс может переопределить правило, ранее заданное в базовом классе. Pega имеет широкое множество достаточно разнообразных фреймворков, на

основе которых строится конечное приложение с использованием наследований, далее приведем для примера некоторые из них:

- PRPC непосредственно является фреймворком;
- DSM — используемый для построения систем принятия решений;
- CPM — используемый для построения систем взаимодействия с клиентурой;
- различные индустриализированные пакеты: страховые, финансовые, медицинские, энергетические и другие.

Основа подхода Pega базируется на трех основополагающих элементах:

- **Situational Layer Cake** (Система слоеного пирога или Слоистая архитектура): за счет полиморфизма и наследования, реализованных в виде RRM (механизма принятия решений), Pega позволяет добиться того, что бизнес-процесс для конкретного клиента изменяется в зависимости от максимально разнообразных условий, в качестве примера, как от уровня лояльности непосредственного клиента, так и от законодательства обслуживающего офиса;[3]
- **6R** - это концепция построения комплексного решения, обеспечивающего получение (Receiving) и назначение (Routing) задач, их отчетность (Reporting), реагирование на данные задачи (Responding), сбор информации (Researching), а также принятие решения и разрешение (Resolving) кейсов;[3]
- **DCO** (Direct Capture of Objectives) - это методология и набор поддерживающих ее средств, направленных на то, чтобы в рамках проекта цели, поставленные бизнесом, реализовывались и захватывались непосредственно.[3]

Pega представляет из себя единое пространство сбора и управления требованиями, для тестирования функционала и разработки, а также для

работы конечного юзера с приложением. При наилучшем возможном сценарии последовательность действий можно представить так:

- выбор первого проекта, который сможет показать наилучший баланс между предполагаемым эффектом и объемом предполагаемых затрат; в дальнейшем мы можем перейти сразу к прочим проектам;
- создание приложения, последующие шаги будут выполняться с использованием средств непосредственного захвата целей (DCO), то есть непосредственно в самой системе;
- фиксирование требований и целей бизнеса;
- внесение спецификаций, описывающих, принципы работы системы, а также связать их с требованиями и/или целями;
- разбиение работы на отдельные подзадачи, по каждой из которых выполнить определенные шаги:
    - провести сессию непосредственного захвата целей (DCO) с предполагаемыми пользователями, чтобы определить единое представление спецификаций, следует помнить, что DCO-сессии служат для верифицирования спецификаций перед началом разработки, а не для сбора требований;
    - реализовать назначенный по данным спецификациям функционал;
    - предоставить результат пользователю.

В подходе Pega однако существует конкретный недостаток: для того чтобы использовать существующие средства DCO существует необходимость создания приложения, для того чтобы непосредственно в нем определять существующие бизнес-цели, а также фиксировать спецификации и прописывать требования. Однако следует помнить, что, при создании приложения существует необходимость указать следующие данные: цели, набор кейсов, бизнес-объекты, базовый фреймворк и необходимая структура

слоев приложения, данная работа подразумевает, что часть задач сбора требований уже была проведена ранее.

Таким образом в ситуации с целями все достаточно нейтрально - для идентификации целей не требуется конкретный инструмент, однако прочие вопросы могут привести к следующему исходу. В первую очередь создается приложение, используемое для сбора требований, а впоследствии создается еще одно приложение, используемое для автоматизации. Впрочем, данный фактор, может быть нивелирован при помощи экспорта и/или импорта спецификаций, а при необходимости и требований.

Далее будет проведен разбор архитектуры IBM. Стек платформ IBM представляет собой набор java enterprise приложений, а не отдельное приложение, и запускается на конкретном сервере «websphere application server». IBM BPM включает в себя следующие элементы:

- бизнес процессное определение - данный элемент представляет из себя диаграмму процесса, создаваемую в соответствии с определенным стандартом, если точнее BPMN. Отдельный процесс имеет возможность включать в себя прочие процессы, а также вызывать отдельный сервис;
- сервис - минимальная возможная к исполнению единица приложения. Структура сервисов подразумевает возможность вкладывать один из них в другой. С точки зрения концепта, сервисы могут быть двух видов: автоматизированные, такие как вызов интеграции или обработка данных, либо неавтоматизированные, то есть требующие прямого обращения к пользователю;
- тип данных - элемент, определяющий структуру данных внутри приложения, также может включать в собственную структуру свойства простых или составных типов, заданных другим тиром данных. Непосредственные переменные и экземпляры объектов данных определяются внутри процесса или сервиса.

IBM BPM включает в себя следующие части:

- репозиторий процессов - хранилище информации о сервисах, процессах, структурах данных и прочем;
- Process center - компонент который обеспечивает доступность репозитория процессов разработки;
- Process server - исполняющий компонент который обеспечивает работу конкретных бизнес-процессов;
- портал - Рабочая среда конкретного пользователя, данный элемент не обязателен;
- Process designer - среда разработки, основанная на eclipse;
- Blueworks Live - облачный сервис для моделирования и документирования конкретных бизнес-процессов. Ранее построенная модель также может быть впоследствии импортирована в process designer, на практике данная возможность используется достаточно редко.
- Process data warehouse - представляет из себя хранилище статистической информации о прохождении экземпляров процессов.

IBM BPM содержит конкретные уровни лицензий:

- Express - содержит полный набор описанных ранее компонентов BPM, однако имеет ограничение в один проект. Может быть установлен только на одиночный сервер, возможности кластеризации отсутствует;
- Standard - содержит полный набор описанных ранее компонентов BPM, не имеет ограничения на количество проектов, возможности кластеризации присутствуют, поддержка интеграции – базовая;
- Advanced - также содержит дополнительную встроенную шину данных и расширенные интеграционные средства. Предназначение набора: высоконагруженные проекты.

IBM также имеет набор сопряженных продуктов, далее приведем некоторые из них:

- ODM - платформа бизнес-правил;
- Case manager - платформа создания и организации набора кейсов;

- Integration bus - интеграционная шина.

Основа подхода IBM заключается в следующем. Несмотря на разницу в методологических подходах данные платформы достаточно схожи по критериям выбора проектов, по определению важности бизнес-целей, а также по подходу к сбору требований и разработке, в IBM значительно меньше инструментов, которые могут быть использованы для поддержки данного подхода.

При внедрении решения на IBM BPM предполагается следующий подход:

- выбор проекта, показывающего наилучший баланс между объемом затрат и потенциальным эффектом;
- описание бизнес-целей в BlueworksLive;
- описание бизнес-процессов в BlueworksLive;
- проведение серии встреч с конечными пользователями, для тестирования процесса ранее описанного в BlueworksLive (Playback);
- перенос процесса из BlueworksLive в Process Center для инициации старта разработки;
- далее итерационно решаются подзадачи:
    - предоставление результатов конечным пользователям (Playback);
    - разработка отдельной части бизнес-процесса.

Таблица 1.1 Сравнение принципов управления требованиями и проектами

PEGA	**IBM**
Принципы управления проектами и требованиями	
Pega предоставляет инструментарий DCO, используемый для управления бизнес-целями и требованиями, определяющими, что должно делаться приложением, и спецификациями, тем каким образом это должно выполняться, с наличием возможности указания связей между ними.	В IBM мы можем вводить цели в BlueworksLive, вводить описания шагов бизнес-процесса, которые являются аналогом спецификации в Pega в BlueworksLive, однако возможность введения общих требований отсутствует, также возможно связи цели со спецификациями или цели с реализацией.

Таким образом мы можем сделать следующие выводы:

- с точки зрения управления проектами и требованиями функционал предоставляемый Pega шире;
- Pega предоставляет инструменты сопровождения процесса разработки, что облегчает взаимодействие между сотрудниками разных групп или направлений.

Далее следует рассмотреть различия в позиционировании между данными платформами. В то время как Pega является обобщенным инструментом для автоматизирования кейсов, бизнес-правил и процессов, а для автоматизирования интеграции предполагается использование внешней шины данных. Также Pega может делегировать отдельные наборы правил или непосредственно правила конечным пользователям, без прерывания процесса разработки этих правил программистом. IBM BPM же предлагается как инструмент для интеграции и автоматизации процессов, в то время как для бизнес-правил и кейсов используются отдельные продукты. Также данная

платформа не может выделять отдельные объекты в эксплуатации без отрыва от производства, однако отдаленной версией данного функционала можно назвать инструментарий IBM ODM, здесь предоставляется возможность раздавать доступ изменения отдельных бизнес-правил различным группам людей.

Рассмотрим основной элемент архитектурные различия платформ. Перейдем к платформе Pega:

- архитектура Pega - объектно-ориентированная. Инкапсулированные объекта данных, логики для обработки данных и интерфейсов ввода-вывода в одном классе является не только рекомендуемым, но скорее обязательным;
- в Pega предполагается, что экземпляр кейса имеет единую область данных, доступную для всех принадлежащих ему шагов, подпроцессов и секций экранных форм. В отдельную область данных выделяют следующие под-кейсы: кейсы настройки маппинга для входных данных для них похожи на аналогичные в IBM, а маппинга для выходных данных агрегируют при помощи отдельного специализированного механизма;
- любое созданное в Pega правило может быть впоследствии пере использовано в прочих приложениях благодаря уникальному механизму наборов правил (Ruleset-ов). Сложность повторного использования дополнительных непредусмотренных изначально компонент - низкая.

Теперь, после ознакомления с основными элементами архитектуры Pega, можно перейти к ознакомлению с архитектурой IBM BPM:

- архитектура IBM в отличии от объектно-ориентированная архитектуры Pega - сервисно-ориентированная (SOA). В IBM невозможно инкапсулированные объекта данных, логики для обработки данных и интерфейсов ввода-вывода в одной единице, что является минусом;
- в IBM BPM каждый отдельный экземпляр процесса или сервиса ограничен областью данных. Для передачи данных от одного процесса к

другому необходимо явно указывать входные и выходные переменные, а также осуществлять их маппинг при каждом вызове;

- в IBM BPM переиспользование отдельных компонент в прочих приложениях возможно только путем выведения их в специальные toolkit-ы, что может потребовать рефакторинга ранее созданных и уже работающих приложений. Сложность пере использования дополнительных непредусмотренных изначально компонент - высокая.

Рассмотрим использующиеся на данных платформах механизмы и техники:

- набор стандартных компонент в Pega неизмеримо больше, чем в IBM BPM, которая обладает достаточно узким набором базовых компонентов. IBM BPM легко дополняет собственные компоненты разработкой новых на языках JavaScript и/или Java, в то время как Pega имеет достаточно узкий набор возможностей кастомизации при помощи чистого кода, что усложняет данный процесс;
- у IBM BPM базово отсутствует механизм адаптации под мобильные интерфейсы, однако подключение дополнительных библиотек позволяет этого достичь, в то время как в Pega подобный механизм адаптации присутствует;
- controls и события в Pega позволяют реализовывать AJAX отправку и автоматическое обновление данных, при помощи выбора соответствующих опций в дизайнер студио. В IBM же обновление данных без перезагрузки формы реализуется значительно сложнее и по отдельности: есть специальные типы сервисов, поддерживающие AJAX-вызовы, а у контролов есть возможность обработки Javascript-событий. Однако, связь событий и сервисов, маппинг данных и обработка результатов выполняется вручную программистом;
- для работы с данными внешних систем используются DataPage, предназначенные для отделения слоя интеграции от слоя данных и слоя бизнес-логики. При использовании данных внешней системы, например, на

экранной форме, в общем случае, не нужно беспокоиться о том, как и когда эти данные были получены. Для операций с данными внешних систем используются специальные типы сервисов. Отделение слоя интеграции от слоя бизнес-логики поддерживается вручную программистами (например, внутренними правилами и рекомендациями по разработке). Для использования данных внешних систем, необходимо либо получить их в качестве входной переменной, либо непосредственно вызывать интеграционные сервисы.

Рассмотрим предполагаемые платформами пользовательские функции:

- в стандартном портале есть социальные функции: обмен сообщениями, информация об участниках процесса. Возможно использование при любых вариантах вызова процесса. В стандартном портале есть социальные функции: обмен сообщениями, информация об участниках процесса, помощь экспертов. Возможно использование только при использовании стандартного портала;
- все почтовые оповещения (о новой задаче, о просрочке и т.п.) настраиваются по отдельности для каждой задачи. Стандартные оповещения в IBM BPM включаются сразу на всё приложение и автоматически оповещают пользователей о всех новых задачах. Однако, оповещения также приходят в случае, если пользователь сам запустил какой-то дополнительный сервис. В связи с этим, в комплексных проектах используются редко;
- стандартных отчетов в Pega больше, чем в IBM BPM, и они дают более полное представление о прохождении процесса. Все дополнительные отчеты реализуются на той же технологии, что и стандартные и размещаются в едином пространстве интерфейса Все дополнительные отчеты реализуются на той же технологии, что и стандартные. Также есть несколько особых отчетов в среде разработки, которые используются для анализа эффективности процесса и прогнозирования результатов любых изменений. Инструмент очень мощный, но требует настроенного сервера разработки и установленной среды разработки. В связи с этим, используется крайне редко;

- порог вхождения для начала работы в Pega Developer Portal: Бизнес-экспертам проще, т.к. на базовом уровне для этого не требуется навыков программирования. Программисту сложнее, т.к. требуется более комплексное понимание архитектуры и удержание в памяти большего числа аспектов. Порог вхождения для начала работы в IBM BPM Process Designer - бизнес-экспертам сложнее, т.к. среда предназначена для программирования. Программисту проще, т.к. для базового уровня достаточно знания небольшого числа специфических элементов.

Далее рассмотрены уникальные преимущества предоставляемые каждой из двух платформ, не имеющие аналога в другой платформе.

Ключевыми уникальными преимуществами Pega можно назвать следующие два элемента: декларативные правила - правила, определяющие конкретные характеристики приложения такие как, к примеру формула расчета бизнес-данных, которые обязаны находиться в процессе выполнения в любой отдельно взятый момент времени. PRPC не опосредованно обеспечивает исполнение и вызов этих правил. Например, правила declare expression, определяющие формулу расчета конкретной переменной.

Выполнение данной формулы движок обеспечивает одним из двух вариантов:

- при каждом изменении каждой отдельной переменной входящей в выражение пересчитывать результат;
- пересчитывать результат только при чтении значения рассчитываемой переменной.

Вторым элементом определяем средства dco: оценивание трудозатрат на проект, прописывание различных уровней документации, налаживание меж целевых взаимосвязей между требованиями и спецификациями.

Далее рассмотрим уникальные преимущества IBM BPM: данная платформа обеспечивает наглядность общего сквозного процесса в BPMN 2.0 (при наличии возможности создания подобного процесса), а также возможности прогнозирования и анализа результатов изменения процесса.

Зачастую не представляет сложности увидеть, как часто экземпляры бизнес-процесса двигаются по одной из веток, при помощи чего посчитать итоговые операционные затраты. После их подсчёта произвести изменения в бизнес-процессе и пересмотреть прогнозируемые изменения в операционных затратах и прочих KPI.

Несмотря на технические сходства и различия обеих платформ, ключевая проблема у данных платформ едина - человеческий фактор, невозможно гарантировать корректное использование функционала платформы. Ключевым элементом успеха большей части проектов являются несколько простых пунктов, по сути независимых от выбранной платформы:

- разработчик должен разбираться в своей платформе: зачастую в реальных проектах на платформах требуется реализовать то, что и так существует, но с учетом конкретно специфики проекта (иначе оформленный портал, собственная система уведомлений и прочее);
- знание особенностей платформы разработки: практически постоянно в различных фактических проектах на представленных платформах необходимо реализовать что-то, что не входит в специфику данной платформы (крайне частый вариант — учетные системы);
- корректная постановка бизнес целей: если ключевой целью реализации проекта является трата бюджета, то успех проекта маловероятен, несмотря на качество выбранной платформы. Однако стоит заметить, что, если проект разрабатывается с пониманием необходимости и эффективности конечной цели, то у команды разработки и заказчиков появляется стимул принимать нужные решения вовремя;
- вовлечение бизнес пользователей в процесс разработки: еще одна зачастую встречаемая проблема заключается в том, что во многих проектах расстояние между бизнес заказчиком и программистом измеряется несколькими руководителями, аналитиками с каждой из сторон, группой архитекторов, группы людей, не вовлеченных в непосредственный процесс и

мегабайтами документации, изменяемой с высокой периодичностью. В следствие чего результат может отличаться от запланированного.

2.3 Разработка концептуальных программно-технических решений приложения.

Для реализации системы уведомления пользователя в специфике медицинского приложения разработана концепция создания типового программного блока для уведомления пользователя в контексте медицинского приложения на платформе Pega.

Концепция определяет цели и задачи создания типового программного блока для решения задач уведомления пользователя на платформе Pega (далее система), общую функциональную архитектуру и состав информационного обеспечения системы, а также этапы ее создания и последующей эксплуатации.

Цель создания системы:

Система создается для решения технической задачи уведомления пользователя о приеме препарата, что обеспечит эффективность используемого приложения и обеспечит его базово-необходимым функционалом.

Для достижения этой цели необходимо обеспечить:

- применение качественной технической базы в виде сервера для расположения на нем платформы разработки и впоследствии используемого для развертывания на нем приложения, доступ пользователей к которому будет обеспечен посредством web интерфейсов;
- разработку базового приложения и его интерфейсов в среде которых и будет работать функциональный блок;
- апробацию, адаптацию и опытную эксплуатацию системы;
- доработку и уточнение состава рабочей документации и проектов нормативной правовой, регламентной и организационной документации.

Основные требования, предъявляемые к системе:

При создании и развитии системы должно быть обеспечено выполнение основных и специальных требований, а также требований к стандартизации и унификации.

Основными требованиями, предъявляемыми к системе, являются следующие:

- системность, состоящая в рациональной декомпозиции системы, в том числе на компоненты и подсистемы системы, предоставляющая возможность автономной разработки и внедрения составных частей системы на основе единой технической политики, что обеспечивает целостность системы при ее взаимодействии с изменяющейся внешней средой;
- открытость, состоящая в способности системы к расширению состава предоставляемых услуг и технологий и увеличению числа источников информации и пользователей без нарушения ее внутреннего функционирования и ухудшения эксплуатационных характеристик;
- стандартизация (унификация), состоящая в рациональном применении типовых, унифицированных или стандартизированных проектных решений и технологий, внутренних и внешних интерфейсов и протоколов, что закладывает фундамент для блочного, модульного построения компонентов и подсистем системы в целом;
- осуществление согласованных между собой процессов проектирования и поэтапной модернизации структурных составляющих системы, обеспечивающих ее постоянную адаптацию к изменяющимся требованиям пользователей.

К специальным требованиям, предъявляемым к системе, относятся:

- полнота информации, обеспечивающей эффективную информационно-аналитическую поддержку пользователей конечного приложения при решении полного комплекса функциональных задач в рамках использования данного приложения;

- комплексная интеграция и рациональное применение при создании системы и ее подсистем существующей базы медицинских данных, типовых решений и технологий;
- мультиплатформенность элементов системы, состоящая в обеспечении возможности функционирования разрабатываемых компонентов системы на любых существующих платформах;
- комплексная безопасность, заключающаяся в осуществлении комплекса мер, призванных обеспечить защиту системы от случайных или преднамеренных воздействий естественного или искусственного характера, связанных с возможностью нанесения ущерба системе и ее пользователям.

Требования к стандартизации и унификации, предъявляемые к системе, включают:

- обеспечение соответствия форматов обмена данными, форматам, используемым во внешних подключаемых сервисах;
- обеспечение возможности централизованного обновления задач системы, при изменении законодательства и методической базы;

Состав и структура системы

Основными компонентами системы являются:

- основной сервер, на котором будет располагаться базовое приложение и который также должен обеспечивать достаточную пропускную способность;
- базовое приложение в среде которого будет существовать основной программный блок;
- внешний сервис смс-гейта, обеспечивающий непосредственное уведомление пользователей.

Для каждой категории пользователей системы реализуются решения по обеспечению безопасности и надежности их автономного и совместного функционирования.

В качестве инструмента работы с системой и обеспечения доступа к содержащимся в ней данным может использоваться специализированное программное обеспечение или веб-браузер.

2.4 Выводы по разделу 2.

Таким образом следует отметить, что в настоящее время рассматривать начальный этап перед разработкой приложения исключительно с точки зрения информационных технологий, без учета интересов бизнеса, текущей ситуации на рынке и прочего не совсем верно, в разрезе существующих в современном мире тенденций. Следует признать, что программист не является отдельным от коллектива сотрудником, он существует в рамках полноценной команды разработки, которая может включать в себя множество специалистов различных направлений: аналитиков, дизайнеров и прочих. Целесообразно рассматривать всю команду как оперативную единицу и подходить к разработке комплексно рассматривая не только ее технические ценность и сложность, но также и востребованность или удобство использования. Следуя из вышесказанного и возникает потребность в определении концепции с учетом которой будет вестись разработка, что позволит специалистам разных направлений представлять своё место в процессе и понимать алгоритмы взаимодействия.

Также стоит обратить внимание на то, что современные рынки переполнены различными платформами, предоставляющими функционал для разработки различного рода приложений, однако помимо того, что каждая из них имеет собственную специфику далеко не все из них конкурентоспособны в среднем. Отдельная платформа может выигрывать в эффективности на конкретном узком направлении, но проигрывать в целом, также следует учитывать и фактор сложности обучения работы на платформе. Таким образом выбор платформы разработки является сложной и важной задачей,

которая определит направление всего последующего процесса, и ошибка на данном этапе может оказаться фатальной для судьбы проекта.

Следующие два критерия стали ключевыми для выбора платформы Pega в качестве основной платформы разработки:

- с точки зрения управления проектами и требованиями функционал предоставляемый Pega шире;
- Pega предоставляет инструменты сопровождения процесса разработки.

Основные требования, предъявляемые к системе разработаны на основании необходимости обеспечения простоты обслуживания, сопровождения и возможного переноса программного блока в прочие продукты.

РАЗДЕЛ 3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА НОТИФИКАЦИОННОЙ ЧАСТИ ПРИЛОЖЕНИЯ MEDINFORM

3.1 Архитектура и описание предварительного этапа кейса уведомлений.

Описание технической части приложения следует начать со следующего замечания, приложение Medinform на определенном этапе разработки было заморожено в связи с переключением на более перспективный и требуемый к выполнению в данный момент проект. Приложение предполагалось, как полноценный сервис обеспечения поддержки и сопровождения пациента, поэтому несмотря на то, что технически данная работа использует нотификационную часть данного приложения, существует определенный набор функций имитация которых была добавлена в предварительную часть кейса нотификации для того, чтобы корректно отобразить основной принцип работы.

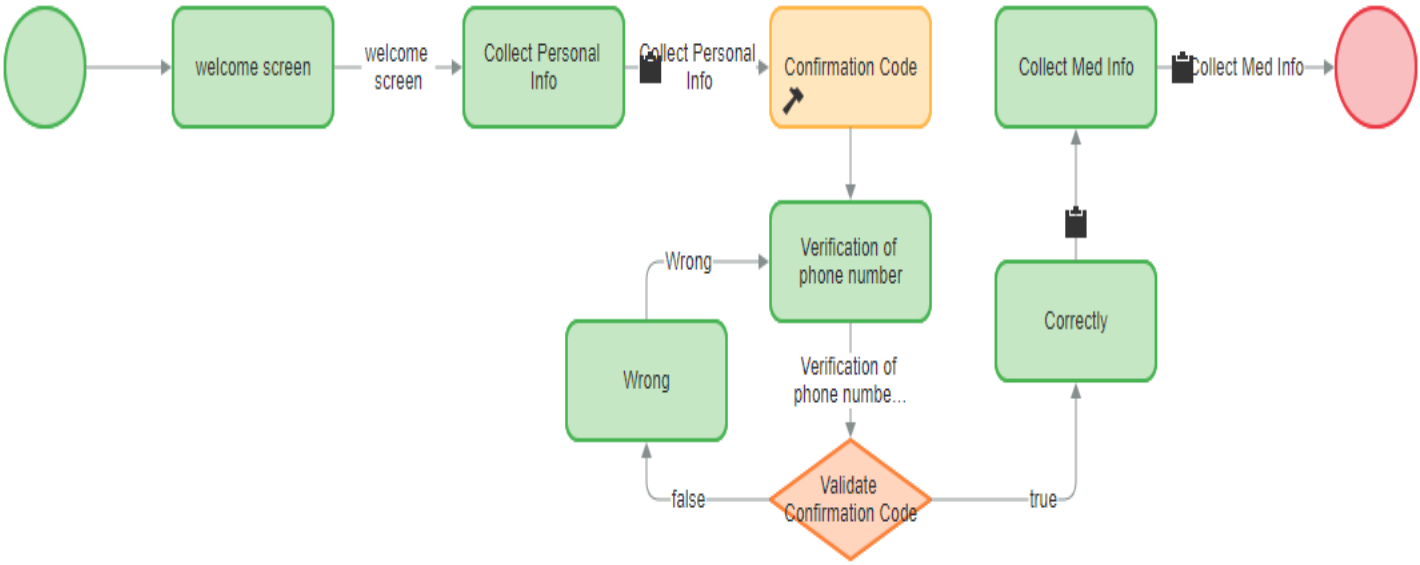


Рисунок 3.1 Схема предварительного процесса сбора информации

Стоит отметить, что данный этап достаточно последователен и предполагает всего одну логическую развилку в ходе своего исполнения. Также стоит отметить понятность и краткость визуального представления

процессов в Пега, что являлось одной из причин выбора данной платформы. Рассмотрим данный процесс пошагово, также дополнительно определив промежуточные процессы, привязанные к коннекторам между отдельных шагов.

Первым шагом данного процесса, является экранная форма, отображающая приветствие, логотип приложения и краткие уточнения для пользователя. Данная форма несмотря на то, что не несет в себе технической необходимости, является достаточно важной с точки зрения юзабилити и дружелюбности интерфейса к пользователю. Более подробно данную экранную форму можно рассмотреть на изображении в приложении Б данной работы. Также на данной форме описаны особенности функционала кейса.

Следующий шаг имитирует собой профиль пользователя, заведенный в системе данного приложения, позволяет ввести контактную информацию пользователя: фамилия, имя, сотовый телефон. Также на данной странице пользователь выбирает удовлетворяющие его требованиям способы уведомления. Визуально данная форма представлена на изображении в приложении Б.

Далее следует блок, состоящий из пяти шагов и двух действий прямой установки данных, привязанных к коннекторам. Первой активируется утилита генерации и отправки кода подтверждения на телефон пользователя. Для значения телефона пользователя дополнительно установлено правило «Edit Validate», код которого представлен в приложении к диплому. Данное активити состоит из пяти последовательных методов:

- создание страницы соединения с сервисом СМС.ру;
- генерация случайного пароля аутентификации для пользователя;
- установка внутренних технических параметров, таких как флаг отправки, ID профиля используемого смс-гейта и прочее;
- запуск преконфигурированного соединения типа RESTс присвоенной ему методой POST;

- закрытие страницы соединения с сервисом СМС.ру.

Далее происходит проверка введенного пользователем пароля, в ходе которой возможны варианты корректного или некорректного ввода, результат проверяется простым булевским выражением, в случае некорректного ввода пользователь перейдет на негативную экранную форму, где ему будет предложено заново ввести пароль, либо повторно его сгенерировать. В случае же корректного ввода пароля пользователь сможет перейти к шагу ввода данных о непосредственно планируемом приеме препарата.

Первым и основным полем данной формы является поле для ввода названия препарата, данное поле, является контролом автокомплит, что обозначает его, как поле для ввода текста с подключенной функцией авто дополнения текста и подключено к внутренней базе данных препаратов, где прописаны определенные препараты с краткой справкой по ним. Также в данном поле установлены свойства из записей базы данных, которые будут выводиться для отображения, а также свойство «имя» установлено как свойство для поиска. На платформе Пега наличествует опция авто генерации кода для свойства авто дополнения.

Далее располагается поле для ввода количества оставшегося препарата, где и как будет использована данная информация будет пояснено в следующей главе. Далее следуют два поля типа дата/время, а конкретнее дата и время первого приема, а также дата и время окончания приема. Здесь также установлена валидация на то, что дата завершения приема стоит во времени позже даты начала приема.

Далее расположен комплексный блок, отвечающий за информацию о выбранном пользователе формате уведомлений. В финальной версии данного кейса было реализовано три формата.

Первый формат, в котором пользователь может получать уведомления: формат по интервалу. В случае выбора данного формата пользователем устанавливается временной интервал: часы, минуты, дни. Сообщения о

приеме препарата будут отсылаться пользователя каждый раз по завершении данного интервала.

Второй формат, несколько более интересен и сложен в разработке основного таймера своей реализации, это формат «по количеству приемов» в день. В случае если пользователь выбирает данный формат ему предполагается указать количество приемов препарата которое он будет совершать в течении одного дня, позже система разобьет доступное в день время приема на соответствующие интервалы самостоятельно и будет отправлять пользователю соответствующее количество сообщений в день равномерно распределяя их га весь период доступного времени.

Третьим же является форма в котором реализована подобная задача во всех без исключения прочих приложениях подобного рода, выбор времени по заданным пользователем значениям. В данном формате пользователь самостоятельно выбирает временные точки дня в которые ему должны приходить сообщения о приеме препарата.

Заключительным блоком данной формы и данного этапа является функциональный, необязательный для использования блок ограничения доставки в ночное время. Данный блок и работа по внедрение его в алгоритмы работы системы уведомлений достаточно уникальны, так как подобное решение не встречалось в ходе рассмотрения медицинских приложений на российском рынке. Суть данного блока проста и заключается в ограничении времени, в котором пользователю могут присылаться уведомления, пользователь самостоятельно определяет со скольких и до скольких приложение может уведомлять его о приеме препарата, стоит отметить что первый и второй форматы по-разному взаимодействуют с данной опцией.

Прежде чем завершить разбор данного этапа рассмотрим, расположенную на данной экранной форме кнопку «добавить препарат», данная кнопка позволяет пользователю инициировать запрос в службу поддержки приложения на добавление нового препарата в систему. Давать

пользователю доступ самостоятельно добавлять препараты в базу данных нелогично, однако существует вероятность, что препарат, который принимает пациент, по той или иной причине отсутствует в базе данных, при помощи данного кейса будет существовать обратная связь между пользователями и разработчиками, что позволит избегать подобных ситуаций. Форма, предполагаемая для заполнения пользователем проста и лаконична и из обязательных полей, предлагает только поле с названием препарата.

3.2 Архитектура параллельных процессов отправки уведомлений.

Этап рассылки уведомлений о приеме препаратов представляет из себя структуру, представленную тремя параллельными процессами, каждый из которых условием своего запуска считывает отдельный формат уведомлений из выбираемых пользователем. Следует заметить, что общая схема каждого процесса достаточна схожа с прочими, а потому далее будет представлен единичный экземпляр схемы.

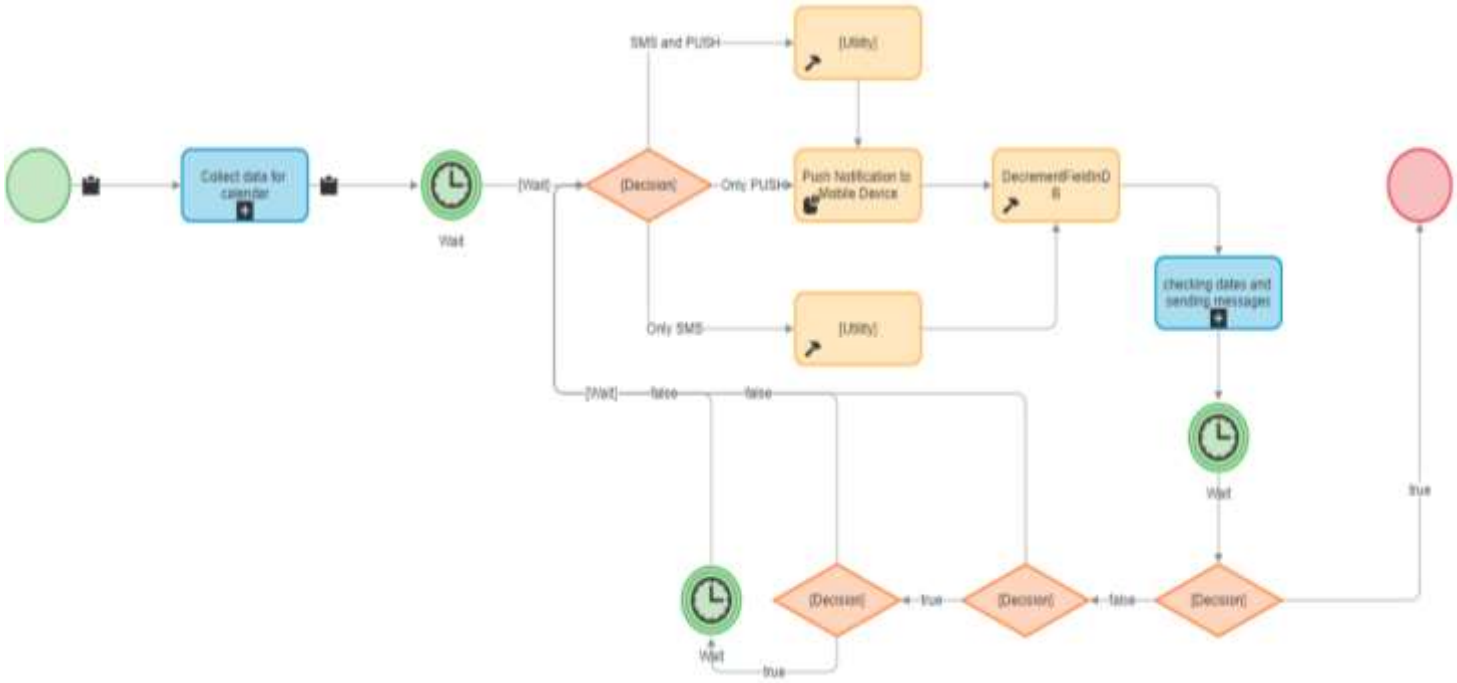


Рисунок 3.2 Общая схема процесса итерационной рассылки уведомлений

Прежде чем перейти непосредственно к описанию данных процессов следует рассмотреть структуру подпроцесса занесения точек приема препарата в базу данных для последующего отображения их на таймлайне по запросу пользователя. Данный подпроцесс также имеет определенные различия в зависимости от выбранного пользователем формата рассылки. Однако существуют общие для всех трех вариантов элементы.

Перед стартом каждого из трех вариантов подпроцесса система на основании ранее введенных пользователем данных генерирует ключевые позиции, такие как идентификатор пользователя для отправки сообщений через сервис CMC.ру или текст получаемого сообщения. Первым действием подпроцесса является занесение записи о приёмах в лист данных приемов.

Далее следуют различающиеся логические блоки, отвечающие за ускоренную имитацию процесса рассылки, для заполнения всех временных точек приема препарата и занесения их в базу, данный процесс был добавлен в приложение после завершения над основными циклами уведомлений и теоретически при определенной доработке и форматизации может представлять альтернативный вариант процесса уведомления пользователя, основанный на записанных в память данных, однако в данной версии программы применяется для иных целей. Наконец финальной частью каждого из подпроцессов является занесение в отдельную базу данных записи об отдельном цикле приема препаратов.

Теперь рассмотрим блоки, отвечающие за итерационную запись данных о приеме препарата, одна из схем подпроцесса занесения точек приема препарата в базу данных для последующего отображения их на таймлайне по запросу пользователя будет представлена далее, две прочие представлены в приложении Б.

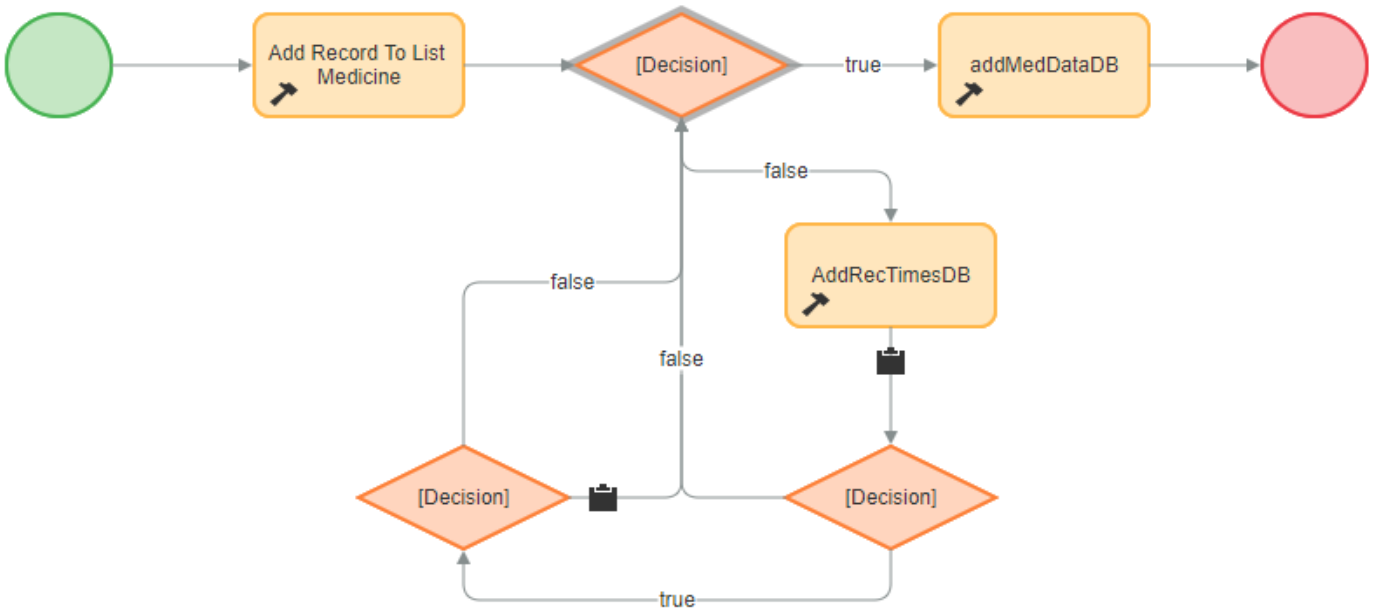


Рисунок 3.3 Схема подпроцесса для процесса отправки по интервалу

Следует отметить, что структура подпроцессов для базовых процессов отправки по интервалу и по количеству приемов имеет визуальное сходства и является значительно более простой нежели чем, схема подпроцесса в случае заданного пользователем времени приема.

В двух данных структурах единственным отличием является внутренняя структура блока решения, отвечающего за переход к следующей временной точке, напрямую следующего из логики данного формата рассылки. В рассылке по интервалу мы сдвигаемся по временной шкале на этот интервал, а в рассылке по количеству мы сдвигаемся на интервал, рассчитываемый делением доступного времени дня на количество приемов. После подсчета каждой точки времени проходит проверка на вхождение в интервал приема, а позднее на отношение текущего условного времени к дате и времени окончания приема.

В случае с подпроцессом для приема препаратов по выбору пользователя, дополнительную сложность, которая находит свое отражение в структуре подпроцесса представляет единичный процесс определения времени первого приема, относительно всех приемов в день, корректное

определение первого интервала и занесения в память временных точек дня первого приема.

Далее происходит переход к основному потоку одного из процессов. Перед входом в основный цикл расположен таймер ожидания времени первого приема, идентичный для всех трех процессов, который сравнивает текущее системное время с занесенным в память временем первого приема препарата.

Затем следует идентичный для всех процессов логический блок из шага принятия решений и набора утилит, отвечающих за фактическую отправку сообщений пользователю. Блок принятия решений инициализирует таблицу решений, которая согласно выбранным пользователем каналам присылает ему сообщения по одному или нескольким из них, следует отметить что данная система самодостаточна, легкодоступна к изменениям и может использоваться для решения прочих подобных задач. Для отправки пуш и емаил сообщений в Pega присутствуют переконфигурированные активити, однако в случае с активити отправки емаил сообщения требуется сконфигурировать настройки почтового ящика, который будет использоваться системой в качестве исходящего, для этого используется инструмент Email Wizard. Окно свойств почтового ящике представлено в приложении на рисунке 3.9

Для отправки смс сообщений отсутствую дефолтные инструменты, поэтому для реализации данной задачи было создано активити состоящее из следующих этапов: инициализация api соединения, установка различных параметров используемых в ходе взаимодействия с смс-гейтом, инициализация REST-сервиса, обеспечивающего фактическую связь с смс.ру и передающего на сайт необходимую для отправки смс сообщения информацию, закрытие api соединения.

Далее следует также идентичный для каждого из трех процессов подпроцесс сверки состояния остатков препарата и количества дней для окончания приема, который инициализирует отправку сообщения, в случае

если у пациента в скором времени закончится препарат или его не хватит на оставшееся время приема, далее мы переходим к самостоятельной для каждого варианта системе таймеров и логических блоков.

В случае отправки по интервалу первый таймер ожидает окончания текущего интервала, после чего логический блок сверяет текущее время с временем окончания приема, если это время еще не наступило и процесс не должен быть завершен, происходят проверки на соответствие времени приема доступному для отправки сообщений времени, далее если пользователь установил возможным прием сообщения в данный момент, процесс вновь переходит к отправке сообщения и цикл повторяется, если же в текущий момент времени сообщение не может быть отправлено запускаюсь таймер ожидания времени рассылки сообщений следующего дня.

В случае отправки по количеству приемов в день первый таймер рассчитывает интервал ожидания в зависимости от количества приемов в день и доступного для отправки сообщений временного промежутка, после чего логический блок сверяет текущее время с временем окончания приема, если это время еще не наступило и процесс не должен быть завершен, происходят проверки на соответствие времени приема доступному для отправки сообщений времени, далее если пользователь установил возможным прием сообщения в данный момент, процесс вновь переходит к отправке сообщения и цикл повторяется, если же в текущий момент времени сообщение не может быть отправлено запускаюсь таймер ожидания времени рассылки сообщений следующего дня.

В случае отправки по заданному пользователем времени первый таймер рассчитывает интервал ожидания в отношении расчета разницы между текущим временем и следующим временем приема в день относительно текущего значения времени, после чего логический блок сверяет текущее время с временем окончания приема, если это время еще не наступило и процесс не должен быть завершен, происходят проверки: на соответствие времени приема на последнюю точку приема в день и на

соответствие времени приема доступному для отправки сообщений времени, далее если пользователь установил возможным прием сообщения в данный момент или точка приема не является последней в день, процесс вновь переходит к отправке сообщения и цикл повторяется, если же в текущий момент времени сообщение не может быть отправлено или данное сообщение должно быть последним в текущий день запускаюсь таймер ожидания времени рассылки сообщений следующего дня.

В завершение следует отметить, что работа с переменными типов дата\время и время связана с определенными трудностями в плане их взаимодействия друг с другом, поэтому в отдельных функциях применялись переменные числовых типов, так как обеспечить их взаимодействие с переменными типа дата\время было проще, а также это не сказывалось на качестве работы системы. Поэтому реализация таймеров для третьего варианта рассылки был связана с определенными трудностями, логика таймеров представлена в приложениях.

3.3 Выбор стороннего сервиса для отправки смс сообщений.

Большинство мобильных приложений встречают конечного пользователя с экрана SMS-авторизации, либо используют смс сообщения в ходе своей работы. Далее мы рассмотрим сервисы смс уведомлений с точки зрения удобства, выгоды использования и надежности его использования для регистрации в мобильных проектах, что в определенной мере отражает безопасность сервиса, отправки смс сообщений или уведомлений, или рассылок и прочих возможных действий.

В данной работе будет представлен сравнительный анализ нескольких СМС-провайдеров или как они называются в большинстве случаев СМС-гейтов. Данная подборка состоит из 3 популярных отечественных сервисов и 3 зарубежных. Использование зарубежного сервиса сопряжено с определенными трудностями, поэтому в результате будет выбран русскоязычный сервис, однако для обеспечения качества сравнительного

анализа будут использованы и зарубежные. Данное сравнение не претендует на максимальную объективность, а отдельные оценки могут оказаться неполными или поверхностным, однако без статистики использования каждого сервиса, сложно подвести объективный результат.

Для обеспечения полноты сравнения, было решено оценить сервисы в различных плоскостях, поскольку занимаемые ими ниши сильно различаются и нацелены данные сервисы, зачастую, на предоставление услуг различного рода.

Приведем список сравниваемых СМС гейтов:

- twilio.com
- plivo.com
- nexmo.com
- smsaero.ru
- sms.ru
- smsc.ru

В первую очередь оценим данные сервисы с точки зрения удобства их интеграции. Начнем с внутреннего структурирования сервисов и их возможностей. Следует заметить, что каждый сторонний сервис, подключаемый к разрабатываемому проекту, нацелен выполнять конкретную задачу, поэтому стоит сосредоточиться на возможностях данных сервисов. Наиболее важный пункт – удобство работы и использования сервиса, а также сложность интеграции его в существующий проект. Было проведено сравнение объема документации сервисов и получены следующие выводы:

Таблица 3.1 Качество документации по сервису

Документация	**Подробная**	**Краткая**
Twilio		+
Plivo		+
Nexmo		+

SMSAero	+	
SMS.ru	+	
SMSC		+

Несмотря на данные таблицы стоит заметить, что англоязычные сервисы, как правило, предоставляют несколько более развернутую документацию, в отличии от российских сервисов СМС-гейтов. В некоторых документациях таких как, документации сервисов Twilio и Nexmo присутствуют наметки на готовые реализации на отдельных языках программирования. В данном разрезе наиболее качественно раскрывается российский SMS.ru, предлагающий развернутые решения насущных SMS-проблем.

Вторым, не менее важным аспектом является базовый функционал представленного сервиса. Для большинства мобильных проектов немаловажно наличие SDK используемого для интеграции, более быстрой и качественной разработки голосовых сервисов и прочих решений. Пусть в нашем случае сервис и будет использоваться только для рассылок уведомлений.

Таблица 3.2 Базовый функционал сервиса

Сервис	**Модули для CMS**	**SDK**	**Песочница**	**Звонки**	**Текст в речь**	**SIP**	**Обратный звонок**	**Рассылка**
Twilio		+						+
Plivo								+
Nexmo	+	+						+
SMSAero			+	+	+	+	+	
SMS.ru			+	+	+	+	+	
SMSC			+	+	+	+	+	

Как видно из представленной таблицы, российский рынок SMS-гейтов в первую очередь ориентирован на массовые рассылки на массовые рассылки сообщений и уведомления пользователей, что и является основным для нас требованием. С другой стороны, западный рынок первоочередно ориентирует платформы на сервисное использование SMS. По сути, мобильный SDK - является фактической диковинкой для представителей данного рынка.

Text-to-Speech (Текст-в-Речь функция) - довольно востребованная функция для авторизации, суть которой заключается в обратном звонке от сервиса, при котором робот проговаривает, например, код авторизации. Данная опция удобна для людей с ограниченными возможностями, а также в случаях, если наблюдаются проблемы с доставкой SMS. Среди проектов ее использующих - Roamer, Telegram и другие. Достаточно интересным нюансом является, что практически все сервисы имеют именно русскую локализацию данной функции, то есть русский голос.

Немаловажную роль играет техническая поддержка и компетентность оказываемой помощи, скорость реагирования на запросы и высокая оперативность при устранении проблемы. Большинство приложений использующих данные сервисы, по своей структуре, успеют столкнуться с технической поддержкой множество раз, поэтому в данной сфере было проведено пристальное исследование предоставляемых услуг, которые выглядят как показано в таблице 3.3.

Таблица 3.3 Техническая поддержка

Сервис	**Чат**	**Форум**	**Почта**	**Скайп**	**Телефон**
Twilio				+	+
Plivo	+			+	+
Nexmo	+			+	

SMSAero					
SMS.ru	+	+		+	
SMSC					

Субъективно следует отметить, что сервисное ориентирование англоязычных сервисов сказывается также и в этой области: российские реагируют несколько быстрее и охотнее, нежели западные конкуренты. В первую очередь это связано с тем, что на действующем отечественном рынке несколько более активно работает процесс продаж данных услуг. Западные сервисы предпочитают отвечать по созданным запросам, в то время как российские могут поддерживать прямой канал связи с пользователем напрямую с сайта.

Самым важным с точки зрения бизнеса являются, тарифные сетки SMS-гейтов, они различны, определенные имеют фиксированные тарифы на заданный объем и не зависят от оператора доставки, некоторые работают по иной схеме — имеют сложносоставную тарифную сетку. Было принято решение отталкиваться от наиболее высоких цен за 1 сообщение, так как специфика российских сервисов и здесь гарантировала серьезные урезания стоимости при увеличении потока сообщения, вплоть до 90% скидок. Интересный факт, отдельные западные сервисы, такие как Twilio, имеют различные тарифы для различных российских операторов.

Все расчеты приведены по ставкам за рассылки с буквенной подписью как показано в таблице 3.4.

Таблица 3.4 Стоимость за 1 сообщение по максимальной ставке

Сервис	**Мегафон**	**Билайн**	**МТС**	**Прочие**
Twilio	1,02 руб.	0,41руб.	0,34 руб.	0,34 руб.
Plivo	0,31 руб.	0,31 руб.	0,31 руб.	0,31 руб.
Nexmo	0,68 руб.	0,68 руб.	0,68 руб.	0,68 руб.

SMSAero	0,65 руб.	0,65 руб.	0,65 руб.	0,65 руб.
SMS.ru	0,69 руб.	0,49 руб.	0,49 руб.	0,25 руб.
SMSC	0,70 руб.	0,70 руб.	0,70 руб.	0,40 руб.

Для полноты картины следует обозначить область доступных платформ оплаты, в данном аспекте западный рынок SMS выглядит значительно более официализированным и менее доступным. Все представленные в выборке англоязычные сервисы не работали с WM, Яндекс.Деньги, а также прочими российскими системами.

Таблица 3.5 Доступные платформы оплаты

Сервис	**ЯД**	**WM**	**Карты**	**Безнал**	**PayPal**
Twilio	+	+			+
Plivo	+	+			+
Nexmo	+	+			
SMSAero				+	+
SMS.ru					+
SMSC					

3.3 Выводы по разделу 3

Таким образом в ходе реализации программного блока были выявлены определенные проблемы и сложности:

- Необходимость получения различной информации необходимой для корректной работы рассылки уведомлений, была решена при помощи предварительного этапа, а более акцентированно его завершающего шага, собирающего набор необходимой информации, в частности комплексный блок, отвечающий за информацию о выбранном пользователе формате уведомлений.

- Необходимость реализации механизма уведомления по выбранному пользователем формату, что было решено через создание трех параллельных процессов, условием выбора одного из которых был выбранный пользователем форма уведомлений.
- Для работы механизма рассылки смс сообщений требуеься подключение стороннего сервиса, однако количество существующих сервисов данной направленности велико и для выбора оптимального варианта был произведен сравнительный анализ существующих предложений.

РАЗДЕЛ 4. ТЕСТИРОВАНИЕ СЕРВИСА НОТИФИКАЦИИ

4.1 Адаптация и тестирование сервиса

Завершенное приложение, не значит правильно работающее приложение. Поэтому приложение должно быть протестировано для определения его работоспособности. Разработка велась в формате концепции SCRUM, что подразумевает итеративный подход к разработке и тестированию продукта. Однако прежде, чем переходить к тестированию следует упомянуть не менее важный момент – адаптацию проекта под условия, в которых он будет использоваться. В ходе изучения различных сервисов уведомлений можно было определить, что адаптация продукта под условия, в которых он будет работать, способна создать значительный разрыв даже между продуктами одного направления.

В данном случае под адаптацией можно понимать такие элементы, как: русификация интерфейсов, приведение экранных форм к привычным и удобным для российского пользователя форматам, а также специфического момента для данного приложения, обеспечения его работы в условиях различных часовых поясов, что требует пусть и минимальной, но настройки приложения. Касательно русификации следует сказать, что различные лейблы и надписи не используемые в программных процессах могут создаваться на русском языке, а техническая информация не требует русификации, однако часть элементов интерфейса создана и сохранена на английском языке, потому требуется использование внутреннего русификатора платформы, он позволяет в таблице сопоставить существующей записи сопоставить ее аналог на русском языке, после чего загрузить таблицу в память приложения, что русифицирует соответствующие элементы.

Относительно процессов тестирования для определения их важности можно определить моменты выявленные именно на этапах тестирования приложения и которые были неочевидны с точки зрения логики процессов.

Одним из таких моментов, на первый взгляд незначительных является следующее: в базовом интерфейсе выполняемого кейса текущему оператору доступна к просмотру таблица со всеми property текущего кейса, а так как код подтверждения являлся на первоначальном этапе разработки переменной кейса, необходимо было ограничить его видимость для пользователя. Одним из вариантов решения данной проблемы могла стать смена типа со значения на параметр кейса, однако в ходе разработки возникло понимание того, что видимость таблицы со значениями в целом не соответствует концепции приложения и перегружает интерфейс для пользователя, поэтому было принято решение изменить внешний вид экранной формы и сделать этот ее элемент недоступным для операторов без прав администратора, так как администратору данная таблица полезна для проверки отсутствия ошибок в записи значений.

Также в ходе тестирования приложения было отмечено, что в случае соответствия времени приема препарата с временем окончания приема пользователю не прислалось сообщение, что представилось нелогичным с точки зрения логики процессов, без четко направленного тестирования данная проблема могла быть не выявлена на протяжение крайне длительного времени и несмотря на кажущуюся незначительность в случае накопления критической массы подобных мелких ошибок, можно было бы говорить о фактической незавершенности продукта.

Платформа Pega предоставляет множество инструментов для тестирования приложений. На основном экране портала студии разработки представлена таблица предупреждений автоматической системы тестирования в разделе на совершенные вами или командой, а также распределенные на три категории по своему типу и критичности. Однако автоматическая система, не определит ошибки подобные вышеуказанным, а также может не указать на конкретный источник ошибки в отдельных случаях. Поэтому в базовый функционал платформы входит расширение Tracer, позволяющее при запуске отслеживать выполняемые процессы

пошагово в разбивке на минимальные системные действия и просматривать всю информацию по этим шагам, при этом система отображает общий статус выполненной операции и цветом акцентирует внимает на ошибках системы.

В последнем обновлении платформы была добавлена новая функция для тестирования кейсов. Внутренние тест кейсы: оператор совершает выполнение кейса и запоминает его в систему впоследствии можно запустить сохраненную прогонку, и система автоматически повторит аналогичный набор действий, что позволит отследить изменения в реакции приложения на идентичные действия до и после внесения каких-либо правок.

4.2 Результаты внедрения и экспериментального использования технического блока

Помимо внутреннего тестирования технического блока, требуется также и тестирование его фактической работы, что предполагает использование функционала рассылки сообщений. В ходе тестирования рассылки проводились следующие этапы:

- тестирование разовой отсылки сообщения в момент достижения таймером времени начала отправки сообщений с таким образом сконфигурированными настройками, что в момент достижения следующей точки рассылки уже будет достигнуто время окончания приема препарата;
- рассылка сообщений в различных конфигурациях, не предусматривающих приближения к моменту, когда препарат в наличии клиента подойдет к минимальному порогу;
- рассылка сообщений в различных конфигурациях, предусматривающих приближение к моменту, когда препарат в наличии клиента подойдет к минимальному порогу;
- рассылка сообщений в пределах различных временных интервалов от суток до месяца.

По результату экспериментального использования технического блока рассылки уведомлений были выявлены специфические для данного продукта особенности. Следует заметить, что в ходе работы данный блок не выдавал сбоев и некорректных или несвоевременных рассылок, однако исходя из самой структуры, так как приложение расположено на сервере количество единовременно обрабатываемых операций ограничено ресурсами данного сервера, что нормально. Также следует заметить, что при проведении дополнительной настройки пользователь может получать сообщения не в виде смс, а в мессенджере Вайбер.

4.3 Выводы по разделу 4

Таким образом можно заключить, что приложение работало в пределах допустимой погрешности в ходе тестирования на наиболее ранних стадиях, которые не подразумевают абсолютной корректности продукта, но на последних этапах разработки, заключавшихся в оптимизации продукта уже, показывало точность и корректность в части рассылки уведомлений.

Следует отметить также и проведенную работу по адаптации и созданию интуитивно понятного интерфейса и максимально лаконичных и информативных сообщений, так как даже в случае идеальной отлаженности и корректной работы приложения некачественные или непонятные для пользователя сообщения или интерфейсы взаимодействия могут оттолкнуть пользователя от конечного продукта.

Также в очередной раз была подтверждена важность тестирования приложения, причем не только при помощи тест-кейсов, но и методами интуитивного тестирования. Даже при максимально качественной подготовке, идеально проработанном техническом задании, работе аналитиков не стоит исключать возможность бага или недоработки в ходе процесса разработки, поэтому тестирование продукта не менее важно, чем

сама по себе разработка, так как сырой продукт неконкурентоспособен в текущих реалиях и не будет пользоваться спросом.

В качестве полезного заключения стоит отметить, что использование внутренних инструментов платформы способно сильно упростить обработку и тестирование приложений, так как предназначено конкретно под разрабатываемое на платформе ПО, в случае с используемой платформой внутренний автоматизированный функционал тестирования позволяет сэкономить значительное количество времени на обнаружение и выявление проблемы при использовании.

ЗАКЛЮЧЕНИЕ

В современных условиях невозможно рассматривать разработку какого-либо приложения отдельно от предварительных этапов планирования и оценки существующей ситуации в предполагаемом сегменте рынка. Рассматривать же отдельный программный блок в отрыве от реалий в которых будет существовать приложение, составной частью которого данный блок является некорректно.

Для обеспечения достаточного уровня качества и соответствия накладываемым требованиям, требуется подходить к непосредственной разработке с наличием понимания того, что требуется от финального продукта. Предварительная подготовка способна сократить временные затраты и обеспечить качественное взаимодействие разработчиков.

Для решения поставленных задач в ходе данной работы было выполнено следующее:

- изучены библиографические источники, предоставляющие необходимую информацию по теме ВКР;
- рассмотрен и изучен рынок приложений медицинской направленности, а также проведено исследование наиболее качественных продуктов;
- изучены методы уведомления пользователя, используемые в подобных приложениях, в результате чего были выбраны наиболее оптимальные в рамках решаемой задачи;
- исследованы возможные платформы разработки, был проведен сравнительный анализ в результате которого была определена платформа разработки;
- изучены концепции о подходе к разработке;
- запрограммирован программный блок при помощи выбранной платформы;

- обеспечено взаимодействие блока со внешними сервисами, посредством уникальных возможностей платформы;
- проведено тестирование блока.

Таким образом все поставленные задачи были выполнены, что свидетельствует о достижении цели работы: создание блока рассылки уведомлений пользователям в разрезе медицинского приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. А.Гилязова. Совершенствование организационно-экономического механизма управления инновациями, 2012. – 74с.
2. Альпина Паблишер. Свод знаний по управлению бизнес-процессами: BPM CBOK 3.0, 2016. – 312с.
3. Андрей Королев. Управление операционной логисической деятельностью, 2016. – 86с.
4. Бартенев В.Г. Программируемая радио электроника – важный фактор иннова ционного обновления России// Современная электроника. № 7. 2010.
5. Бертран Мейер. Почувствуй класс. Учимся программировать хорошо с объектами и контрактами, 2015. – 324с.
6. Вон Вернон. Implementing Domain: Driven Design. 2017 – 688с.
7. Грачев И. В. Методика поддержки модельно- ориентированного процесса разработки программного обеспечения // Материалы Всерос. науч.-практ. конф. «Системы промышленного и информационного сервиса (инфраструктура, 5. объекты, процессы)». Кострома, 2008.
8. Грегор Хоп, Бобби Вульф. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. 2016 – 672с.
9. Гусев Д. И., Коротеева О. С. О развитии информационных технологий в системе управления организации // Интеграл. 2012. № 5.
10. Джон Джестон. Управление бизнес-процессами. Практическое руководство по успешной реализации проектов, 2012. – 125с.
11. Е.Б.Грибанова. Процессно-ориентированное моделирование систем массового обслуживания, 2015. – 83с.
12. Импелтех. Графический язык моделирования бизнес-процессов BPMN Версия 2.0, 2016. – 298с.
13. Кент Бек. Implementation Patterns. 2017 – 176с.

14. Костров А. В., Коротеева О. С., Корнюшко В. Ф. Особенности информационного менеджмента в компаниях сферы услуг // Прикладная информатика. 2012. № 1.

15. Костров А. В., Коротеева О. С., Якунченко-ва С. Ю. Оценка уровня развития информационного менеджмента // Прикладная информатика. 2012. № 3.

16. Костров А. В., Полянский Е. И. Обоснование обобщенных критериев оценки распределенной информационной системы на основе морфологического анализа // Интеграл. 2012. № 3.

17. Кузин Р. Е., Кожин О. В., Лебедев И. В., Моги-рев А. М, Писаненко С. С., Таиров Т. Н. Система информационной поддержки радиационного контроля большого потока проб // Прикладная информатика. 2012. № 2. С. 26-31.

18. Моделирование систем, Объектно-ориентированный подход // Колесов Ю., Сениченков Ю., 2012. – 364с.

19. Стив Макконнелл. Code Complete. 2007 – 896с.

20. Управление внедрением модельно-ориентированного подхода в процесс разработки программного обеспечении // С.С. Писаненко, И.В. Грачев, О.А. Жданович, А.А. Тимофеев 2013 – 86с.

21. Bartenev V. Software Radar: New Reality. Report on the International conference RADAR 2006. China. 2006.

22. Henrik Kniberg.Скрам и XP заметки с передовой.: Пер. с Англ. Agile Ukraine, 2012. – 94с.

23. Magic Quadrant for BPM-Platform-Based Case Management Frameworks // Gartner: Janelle B. Hill, Kenneth Chin, Rob Dunie – 2015. – 12 с.

24. Model-Based Design. URL: www.mathworks.com. (дата обращения: 27.04.2018).

25. MSC Software. Extending Simulation to the Enterprise. 2010. URL: http://www.mscsoitware.ru/products/nastran3 (дата обращения: 21.04.2018).

26. PegasystemsInc., Cambridge,MA Senior System Architect Essentials 7.2 StudentGuide, 2017. – 450c.

27. PegasystemsInc., Cambridge,MA Senior System Architect Essentials 7.2 Exercise Guide, 2017. – 219c.

28. PegasystemsInc., Cambridge,MA System Architect Essentials 7.2 Exercise Guide, 2017. – 302c.

29. PegasystemsInc., Cambridge,MA System Architect Essentials 7.2 StudentGuide, 2017. – 443c.

30. URL: www.microchip.com. (дата обращения: 28.04.2018).

ПРИЛОЖЕНИЕ А. Сравнительная таблица PEGA и IBM BPM

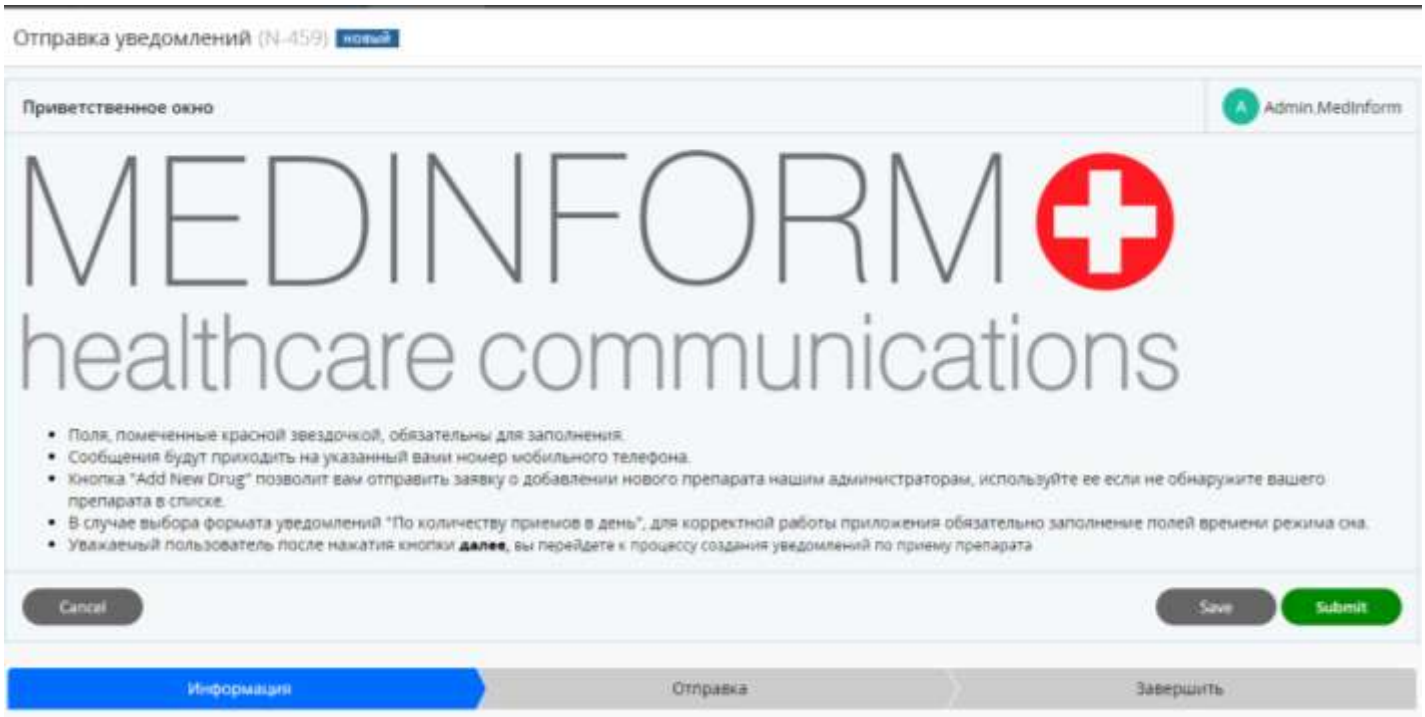
PEGA	**IBM**
Управление требованиями и проектом	
Pega предоставляет интрументы DCO для управления бизнес-целями, требованиями (ЧТО приложение должно делать) и спецификациями (КАК приложение должно работать) с возможностью указания связей между ними. Также со спецификациями можно связать результаты разработки, чтобы отслеживать прогресс проекта.	В IBM есть возможность ввода целей в BlueworksLive, ввода описаний шагов бизнес-процессов (аналог спецификации в Pega) в BlueworksLive, но нет возможности ввести общие требования, связать цели со спецификациями или с реализацией.
Различия в позиционировании	
Pega это единый инструмент для автоматизации кейсов, процессов и/или бизнес-правил. Для автоматизации интеграции рекомендуется использовать внешнюю шину данных.	IBM BPM это единый инструмент для автоматизации процессов и интеграции. Для кейсов и бизнес-правил в стеке IBM есть отдельные продукты.
Pega имеет возможность делегирования отдельных правил бизнес-пользователям в то время, пока над остальными работают программисты.	IBM BPM не имеет возможности отделять часть правил и/или бизнес-процесса, чтобы их могли изменять пользователи во время “боевой” эксплуатации, но подобный инструмент есть в IBM ODM: там есть возможность предоставлять доступ на изменение отдельных бизнес-правил разным группам людей.
Архитектурные различия	
Pega имеет объектно-	IBM имеет сервисно-ориентированную

ориентированную архитектуру. В ней возможно и рекомендуется инкапсулировать в одном классе объект данных, логику обработки этих данных и интерфейсы для ввода и вывода.	архитектуру (SOA). В нем невозможно инкапсулировать в одной единице объект данных, логику обработки этих данных и интерфейсы для ввода и вывода.
В Pega экземпляр кейса имеет единую область данных, доступную в т.ч. для всех его шагов, подпроцессов и секций экранных форм. Отдельную область данных имеют под-кейсы: настройка маппинга входных данных для них похожа на аналогичную в IBM, а выходные данные агрегируются через отдельный специальный механизм.	В IBM BPM каждый экземпляр процесса или сервиса ограничен своей областью данных. Для передачи данных от одного процесса к другому (например, при открытии экранной формы с данными процесса) необходимо явно указывать входные и выходные переменные и осуществлять их маппинг при каждом вызове.
Pega предоставляет возможность переиспользования любого созданного правила в другом приложении благодаря механизму Ruleset-ов. Сложность переиспользования дополнительных непредусмотренных изначально компонент: низкая.	В IBM BPM переиспользование отдельных компонент в другом приложении возможно только путем выделения их в специальные toolkit-ы, что может потребовать рефакторинга уже созданных и работающих приложений. Сложность переиспользования дополнительных непредусмотренных изначально компонент: высокая.
Использование техник и механизмов	
Pega имеет небольшой набор возможностей кастомизироваться хард-кодом, но обладает очень широким набором стандартных компонент.	IBM BPM обладает существенно меньшим набором стандартных компонент, но позволяет легко разработать собственные на языках JavaScript и/или Java.
В Pega есть встроенный механизм адаптации интерфейса под	У IBM BPM из коробки пока нет механизма адаптации под мобильные

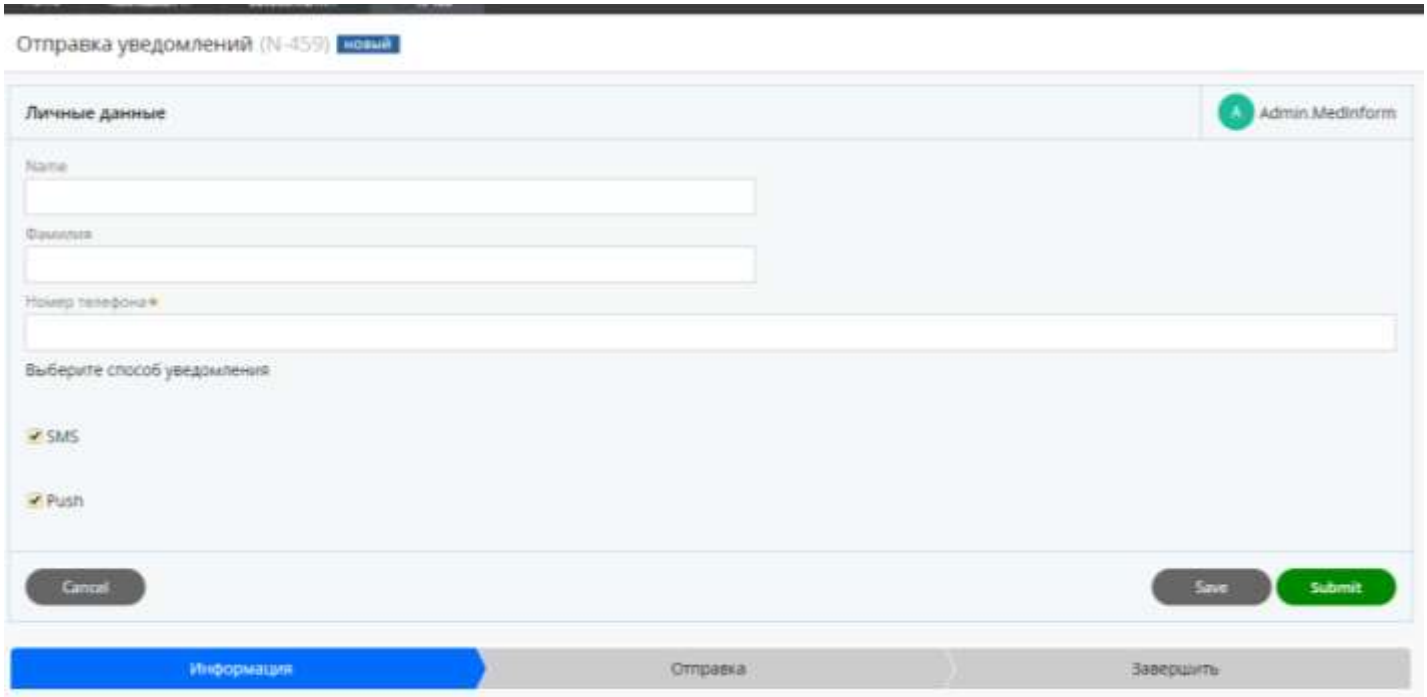
мобильные устройства.	интерфейсы, но есть дополнительная подключаемая библиотека (toolkit), позволяющая этого достичь.
Контролы и события в Pega позволяют реализовать AJAX отправку и обновление данных автоматически, путем выбора соответствующих опций в дизайнере.	В IBM обновление данных без перезагрузки формы реализуется по отдельности: есть специальные типы сервисов, поддерживающие AJAX-вызовы, а у контролов есть возможность обработки Javascript-событий. Однако, связь событий и сервисов, маппинг данных и обработка результатов выполняется вручную программистом.
Для работы с данными внешних систем используются DataPage, предназначенные для отделения слоя интеграции от слоя данных и слоя бизнес-логики. При использовании данных внешней системы, например, на экранной форме, в общем случае, не нужно беспокоиться о том, как и когда эти данные были получены.	Для операций с данными внешних систем используются специальные типы сервисов. Отделение слоя интеграции от слоя бизнес-логики поддерживается вручную программистами (например, внутренними правилами и рекомендациями по разработке). Для использования данных внешних систем, необходимо либо получить их в качестве входной переменной, либо непосредственно вызывать интеграционные сервисы.
Пользовательские функции	
В стандартном портале есть социальные функции: обмен сообщениями, информация об участниках процесса. Возможно использование при любых вариантах вызова процесса.	В стандартном портале есть социальные функции: обмен сообщениями, информация об участниках процесса, помощь экспертов. Возможно использование только при использовании стандартного портала.
Все почтовые оповещения (о новой	Стандартные оповещения в IBM BPM

задаче, о просрочке и т.п.) настраиваются по отдельности для каждой задачи.	включаются сразу на всё приложение и автоматически оповещают пользователей о всех новых задачах. Однако, оповещения также приходят в случае, если пользователь сам запустил какой-то дополнительный сервис. В связи с этим, в комплексных проектах используются редко.
Стандартных отчетов в Pega больше, чем в IBM BPM, и они дают более полное представление о прохождении процесса. Все дополнительные отчеты реализуются на той же технологии, что и стандартные и размещаются в едином пространстве интерфейса.	Все дополнительные отчеты реализуются на той же технологии, что и стандартные. Также есть несколько особых отчетов в среде разработки, которые используются для анализа эффективности процесса и прогнозирования результатов любых изменений. Инструмент очень мощный, но требует настроенного сервера разработки и установленной среды разработки. В связи с этим, используется крайне редко.
Порог вхождения для начала работы в Pega Developer Portal: Бизнес-экспертам проще, т.к. на базовом уровне для этого не требуется навыков программирования. Программисту сложнее, т.к. требуется более комплексное понимание архитектуры и удержание в памяти большего числа аспектов.	Порог вхождения для начала работы в IBM BPM Process Designer: Бизнес-экспертам сложнее, т.к. среда предназначена для программирования. Программисту проще, т.к. для базового уровня достаточно знания небольшого числа специфических элементов.

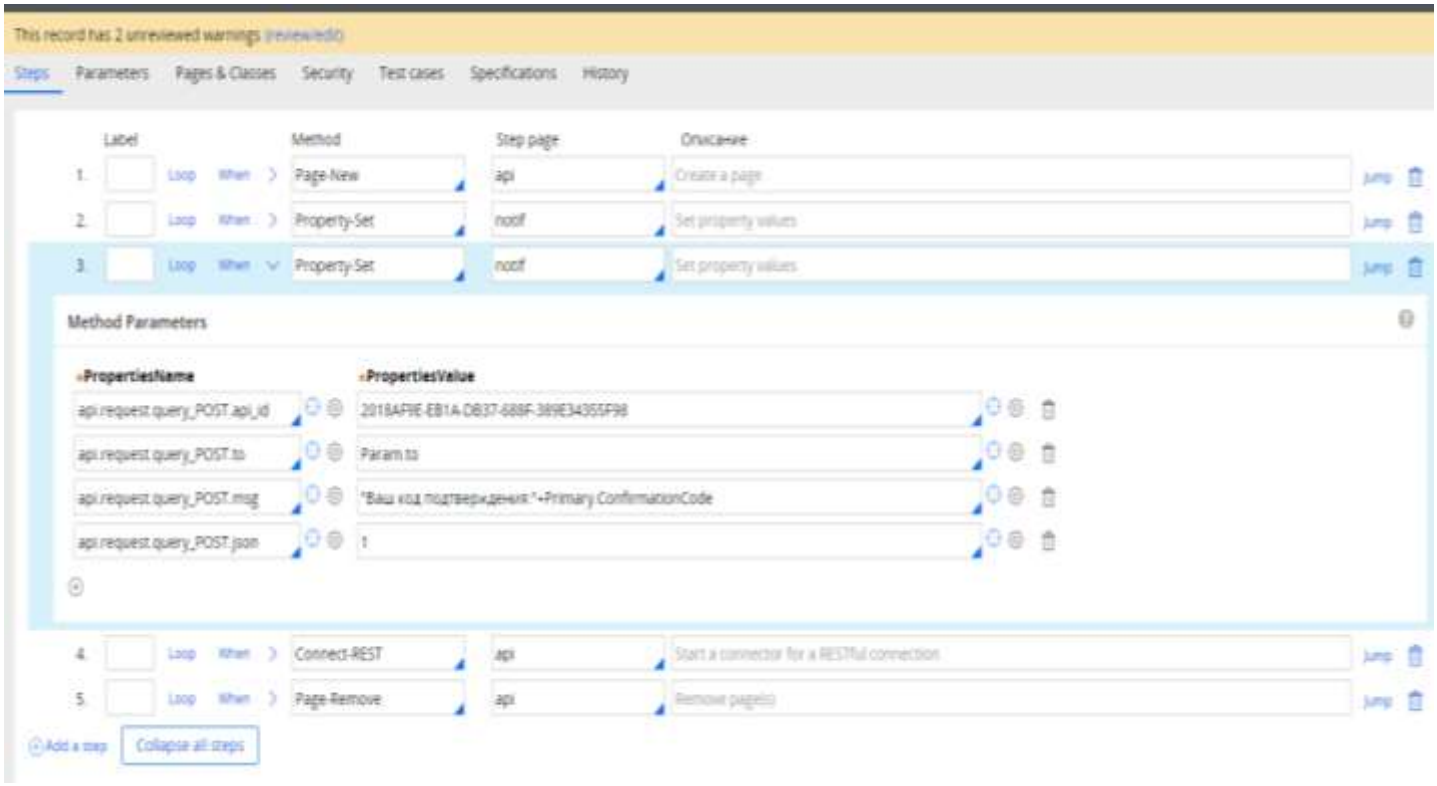
Приложение Б. схемы и экранные формы



Экранная форма приветствия



Контактная информация



Основная страница активити генерации проверочного кода

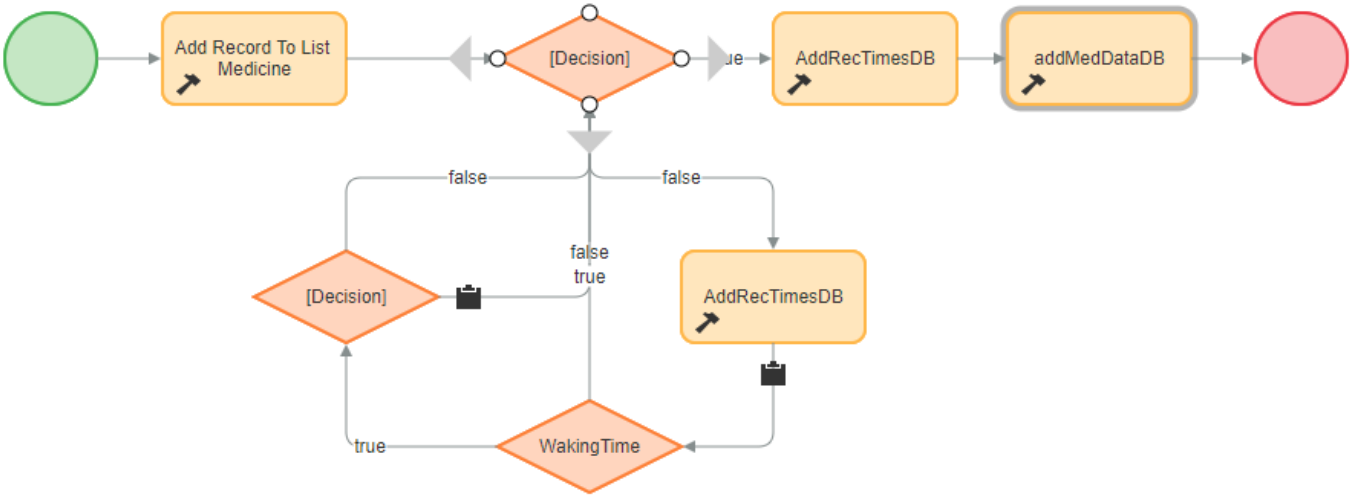


Схема подпроцесса для отправки по количеству в день

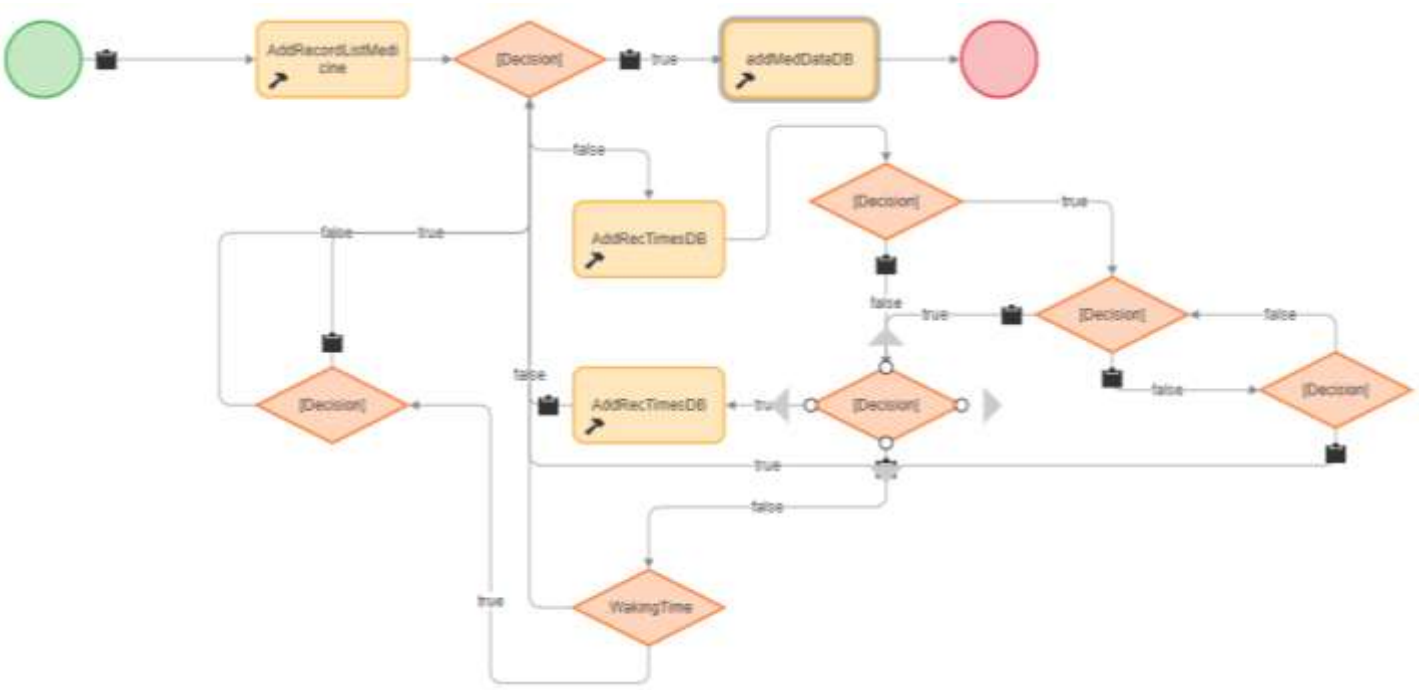
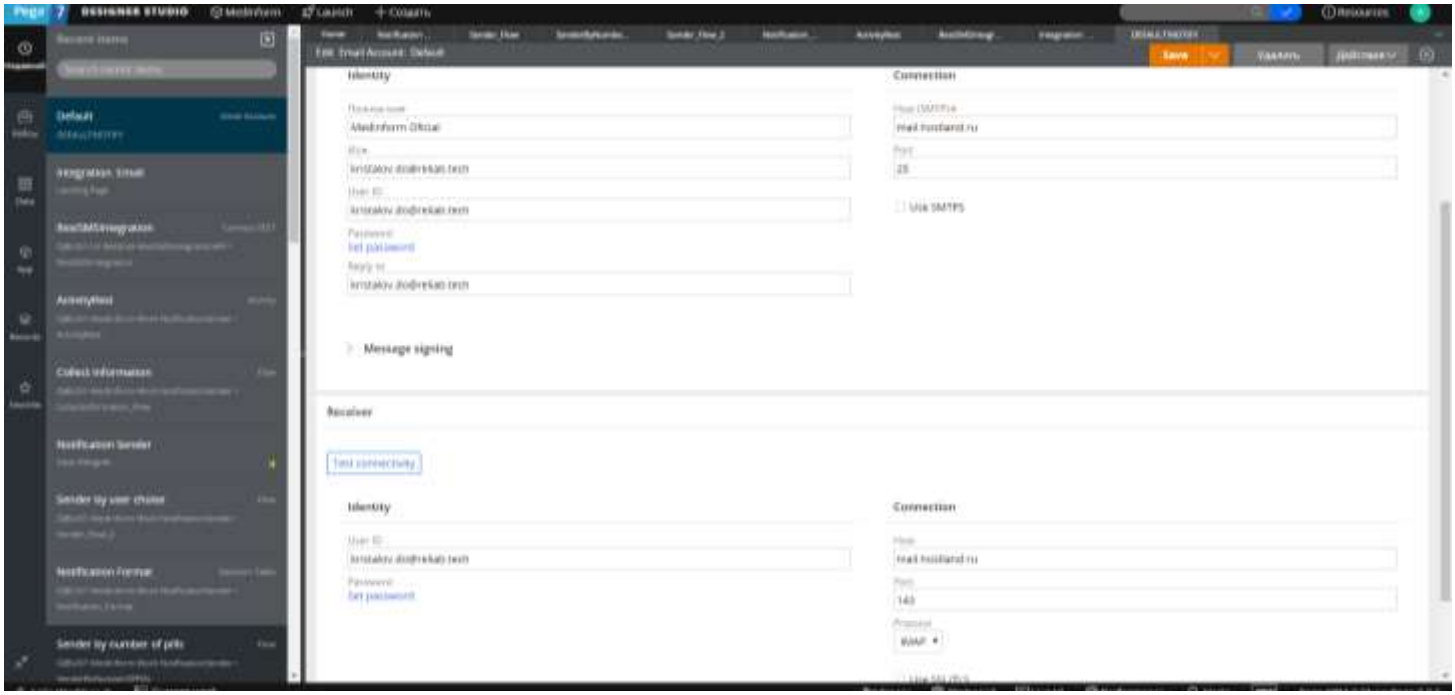


Схема подпроцесса для отправки по заданному пользователем времени



Свойства основного почтового ящика

Приложение В. Элементы кода

Основные таймеры для всех трех вариантов рассылки приложений в порядке, определенном их упоминанием в тексте диплома:

1. @addToDate(@CurrentDateTime(),.Interval.Days,.Interval.Hours,.Interval.Minutes,0)
2. @addToNow(0,0,0,((((toInt(.DownTimeHours)) - (toInt(.UpTime))) * 3600 + (toInt(.DownTimeMinutes)) * 60 - (toInt(.UpTimeMinutes)) * 60)/.NumberOfReceptions))
3. addToNow(0,(toInt(.CustomReceptionTimes(.K).Hours)-toInt(stripCharsOffEnd(getCurrentTimeOfDayOnlyStamp(),4))),(toInt(.CustomReceptionTimes(.K).Minutes) - ((toInt(stripCharsOffEnd(getCurrentTimeOfDayOnlyStamp(),2)))-toInt(stripCharsOffEnd(getCurrentTimeOfDayOnlyStamp(),4))*100)),0)

Код проверки соответствия введенного значения заданной маске:

```
String RUPhone = "^\\+?([7]{1})?\\(?[0-9]{3}\\)?[0-9]{3}\\-?[0-9]{2}\\-?[0-9]{2}$";
java.util.regex.Pattern p = java.util.regex.Pattern.compile(RUPhone);
java.util.regex.Matcher m = p.matcher(theValue);

boolean isRUPhone = m.matches();
if((isRUPhone == false) ){
	return false;
}
if(theValue.length()!=11) {
	return false;
}
return true;
```