Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

ПРОЕКТ ПРОВЕРЕН
Рецензент
Руководитель отдела QA ООО
«Инфиннити»
_____ В.М. Орлов
«___» _____ 2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
«Информационные технологии в
экономике», д.т.н, с.н.с.,
_____ Б.М. Суховилов
«___» _____ 2018 г.

Автоматизация тестирования мобильных приложений на примере разработки сервиса «Классная «Москва»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОМУ КВАЛИФИКАЦИОННОМУ ПРОЕКТУ
ЮУрГУ – 09.03.02.2018.231 ПЗ ВКП

Консультант
по экономической части работы,
старший преподаватель
_____ А.Г. Шепталин
«___» _____ 2018 г.

Руководитель работы, к.п.н.
доцент
_____ С.А. Тимаева
«___» _____ 2018 г.

Консультант
по технической части работы,
доцент
_____ Б.В. Иваненко
«___» _____ 2018 г.

Автор работы
студент группы ЭУ-489
_____Е.И. Черкасов
«___» _____ 2018 г.

Нормоконтролёр, к.п.н.
доцент
_____ С.А. Тимаева
«___» _____ 2018 г.

Челябинск 2018

АННОТАЦИЯ

Черкасов Е.И. Автоматизация тестирования мобильных приложений на примере разработки сервиса «Классная «Москва» – Челябинск: ЮУрГУ, ЭУ-489, 75 с., 30 ил., 12 табл., библиогр. список – 9 наим., 1 прил.

В работе описано исследование ООО «Инфиннити» и проект автоматизации тестирования мобильных приложений.

Проанализировано дальнее и ближнее внешнее окружение предприятия, и его влияние на работу организации.

Рассмотрены бизнес-процессы предприятия. В работе выявлены слабые и сильные стороны организации, угрозы и возможности внешней среды.

Описан процесс тестирования мобильных приложений. Найдены проблемы, связанные с данным процессом.

Описан проект внедрения системы, позволяющий решить проблему компании.

Разработан интерфейс внедряемого приложения, написан его код на языке Python, описан успешный сценарий его работы, заявлены требования к техническим средствам.

Разработан инвестиционный проект внедрения системы, проанализированы риски. Проведен анализ экономической эффективности инвестиционного проекта.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы. В последние годы активно развивается рынок мобильных устройств и приложений. При тестировании таких приложений возникают трудности, связанные с большим количеством различных версий мобильных устройств.

Автоматизация такого тестирования позволит сократить время на процесс ручного тестирования, позволит повысить качество самого тестирования и, как следствие, приложения в целом.

Цель: автоматизация тестирования мобильных приложений в ООО «Инфиннити».

Задачи:

1. Проведение предпроектного обследования компании и сервиса;
2. Выявление существующих проблем в организации;
3. Выделение требований к будущей системе;
4. Проведение анализа рынка готовых решений программного обеспечения;
5. Разработка программного обеспечения;
6. Проектировка интерфейса пользователя;
7. Разработка проекта для внедрения;
1. Оценка эффективности проекта.

Объект исследования: компания ООО «Инфиннити».

Результаты работы рекомендуется использовать при тестировании мобильных приложений в ООО «Инфиннити».

ГЛАВА 1. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ БИЗНЕСА

1.1 Общие сведения о предприятии

Основным направлением деятельности компании «Инфиннити» является разработка информационных систем, бизнес веб-приложений и корпоративных программных решений, используемых в производственной, коммерческой и социальной сферах. Технологическая основа для разработки как собственных тиражируемых продуктов, так и заказных решений – платформенное программное обеспечение InfinniPlatform [2].

1.1.1 История предприятия

Компания ООО «Инфиннити» основана в 2006 году. Генеральный директор – Гребнев Павел Евгеньевич. В том же году «Инфиннити» разработала программный продукт Медицинская информационная система «МЕДИК+» для решения актуальных задач и сокращения бумажного документооборота в поликлиниках Челябинской области.

В 2009 году решением Координационного совета Минздрава Челябинской области в 2009 году МИС «МЕДИК+» была утверждена, как типовая система для ЛПУ и рекомендована для безвозмездной передачи и внедрения в учреждения здравоохранения Челябинской области.

В 2010 году по заданию Челябинского областного МИАЦ МИС «МЕДИК+» была доработана до последних требований Минздравсоцразвития РФ, включая требования законодательства по защите персональных данных при ведении медицинских карт в электронном виде. Для централизованного распространения и доступа к МИС посредством тонкого клиента через веб-интерфейс для поликлиник Челябинской области была выпущена подсистема «Электронная регистратура».

В 2012 году совместно с компаниями IBS и Ocean Informatics выигран конкурс на разработку и пилотное внедрение ИЭМК для Департамента Информационных Технологий г. Москва. По заданию городского управления

здравоохранением г. Челябинска развернута общегородская система записи на прием ко врачу через Интернет.

1.1.2 Характеристика предприятия

Компания «Инфиннити» производит следующие виды продуктов:

- Информационные системы
- Компоненты информационных систем
- Web-приложения;
- Корпоративные приложения;
- Мобильные приложения;

1.2 Цели предприятия

1.2.1 Миссия

Миссия компании «Инфиннити» – «Мы стремимся создавать современные, инновационные, качественные программные продукты, которые смогут удовлетворить потребности клиентов.»

1.2.2 Стратегическая карта целей

Стратегическая карта (Рисунок 1.1) – это элемент системы сбалансированных показателей. Представляет собой диаграмму, на которой обозначены основные цели существования организации. Цели на карте связаны между собой направленными причинно-следственными связями. Связи позволяют проследить воздействие одной цели на другую.

Цели «Инфиннити» разделены на 4 ключевых пространства: «обучение и развитие», «бизнес-процессы», «клиенты», «финансы». Каждая цель имеет показатель, позволяющий отслеживать развитие компании. Улучшение или ухудшение показателя указывает на развитие или снижение темпов развития.

Основной целью компании «Инфиннити» является «Повышение прибыли», достичь этого можно несколькими способами:

- Привлечение новых клиентов. Ведет к увеличению доходов.
- Удержание клиентов. Ведет к увеличению доходов.
- Повышение эффективности производства продуктов. Ведет к уменьшению расходов.
- Повышение эффективности производства услуг. Ведет к уменьшению расходов.

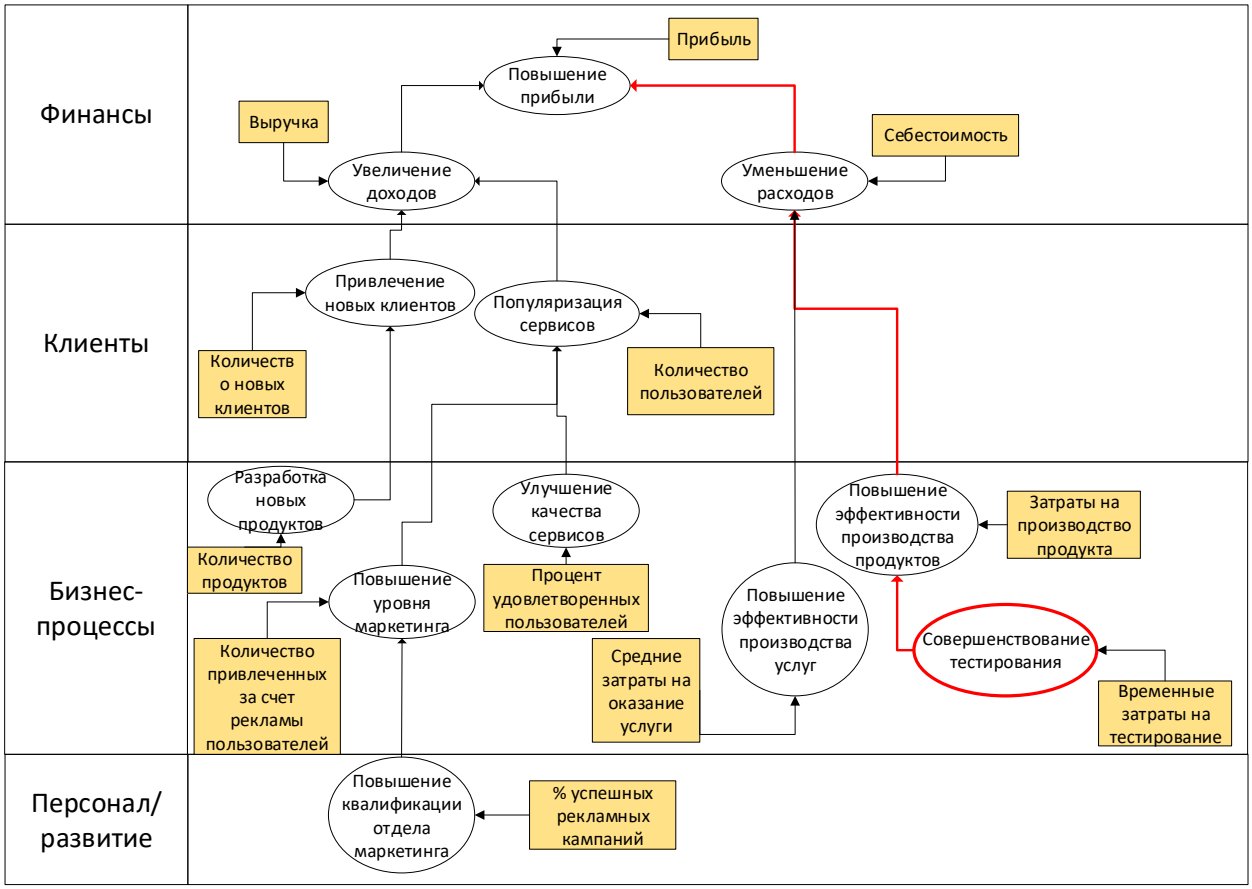

Рисунок 1.1 – Стратегическая карта целей

Счетная карта приведена в таблице 1.1.

Таблица 1.1 – Счетная карта

Область	Цель	Показатель	Ед. изм.	Критерий достижений целей
Финансы	Повышение прибыли	Прибыль	Млн. руб.	Прибыль ≥ 20 млн. руб.
	Увеличение доходов	Выручка	Млн. руб.	Выручка ≥ 50 млн. руб.
	Уменьшение расходов	Себестоимость	Млн. руб.	Себестоимость ≤ 30 млн. руб.

Окончание таблицы 1.1

Клиенты	Привлечение новых клиентов	Количество новых клиентов	шт.	Количество новых клиентов ≥ 10
	Популяризация сервисов	Количество пользователей	Тыс. шт.	Количество пользователей ≥ 20 тыс. шт.
Область	Цель	Показатель	Ед. изм.	Критерий достижений целей
Бизнес-процессы	Разработка новых продуктов	Количество продуктов	Шт.	Количество продуктов ≥ 15 шт.
	Повышение уровня маркетинга	Количество привлеченных за счет рекламы пользователей	Тыс. шт.	Количество привлеченных за счет рекламы пользователей ≥ 5 тыс. шт.
	Улучшение качества сервисов	% удовлетворенных пользователей	%	% удовлетворенных пользователей ≥ 90%
	Повышение эффективности производства услуг	Средние затраты на оказание услуги	Тыс. руб.	Средние затраты на оказание услуги ≤ 1500 тыс. руб.
	Повышение эффективности производства продуктов	Затраты на производство продукта	Тыс. руб.	Затраты на производство продукта ≤ 2500 тыс. руб.
	Совершенствование тестирования	Временные затраты на тестирование	Чел/час	Временные затраты на тестирование ≤ чел/час
Персонал / Развитие	Повышение эффективности отдела маркетинга	% успешных рекламных кампаний	%	% успешных рекламных кампаний ≥ 80%

1.3 Анализ дальнего окружения

Каждая организация существует в определенной внешней среде, которая влияет и формирует ее структуру, создает положительные возможности и нежелательные обстоятельства, влияние которых различно, вследствие чего выделяют факторы микросреды (ближнее окружение) и макросреды (дальнее окружение). Для разработки устойчивой стратегии бизнеса предприятию необходимо проводить анализ этих факторов [9].

Для анализа дальнего окружения предприятия воспользуемся STEP-анализом.

Цель: Выявить факторы, наибольшим образом влияющие на деятельность предприятия, отследить их тенденцию и сделать прогнозный характер внешней среды.

Факторы:

Социальные(S):

- Возрастание требования потребителей к качеству приложений.

  Возрастают потребности общества в качественных мобильных, десктопных и веб-приложениях. Это требует более серьезного подхода к разработке приложений и систем.

- Популяризация социальных сетей.

  Все больше людей начинает пользоваться социальными сетями. Возрастает спрос на приложения с элементами социальных сетей.

Технологические(T):

- Развитие прикладного, базового программного обеспечения для разработки информационных систем.

  Разрастается рынок программного обеспечения для разработки информационных систем, а также развиваются технологии проектирования. Кроме коммерческих продуктов, популярность обретает разработки свободного программного обеспечения. Такая динамика позволяет разработчикам выбирать средства проектирования из множества вариантов, использовать те технологии, которые более применимы к конкретному проекту.

- Развитие облачных технологий.

  Отрасль облачных технологий активно развивается. Появляются новые решения, больше потребителей узнают о достоинствах работы через облако.

- Появление большого количества мобильных устройств с разными конфигурациями аппаратного и программного обеспечения.

Рынок смартфонов активно развивается. Появляются новые технологии, изменяются функциональные особенности текущих операционных систем, а также эти системы дополняются различными надстройками. Поэтому возникает необходимость оптимизировать приложения для многих версий устройств.

Экономические(Е):

- Экономический кризис.

  Неблагоприятная экономическая обстановка отрицательно сказывается на желании фирм разработать систему «под себя». Фирмы не хотят идти на риски и либо совсем отказываются от внедрения информационных систем, либо выбирают уже готовые и проверенные решения.

- Возрастание доли финансовых вложений со стороны государства.

  В настоящее время государство активно занимается модернизацией различных сфер, в том числе здравоохранения. Поэтому повышение интереса со стороны государства к совершенствованию систем приводит к большему финансовому вливанию в эти бюджетные организации. Это позволяет им направлять эти денежные средства в автоматизацию деятельности.

Политические(Р):

- Федеральный закон №152 «О персональных данных».

  Закон «О персональных данных» и сопутствующие ему постановления накладывают на оператора персональных данных обязательства по приведению инфраструктуры и бизнес-процессов системы к состоянию, при котором будет обеспечена конфиденциальность персональных данных. Системы должны пройти процедуру сертификации, компании-разработчики должны получить необходимые разрешающие документы.

- Ухудшение отношений с Америкой.

В последние годы отношения между Россией и Америкой ухудшаются. Поскольку среди партнеров компании «Инфиннити» есть американские компании, это может привести к потере данных партнеров.

STEP-анализ дальнего окружения «Инфиннити» приведен в таблице 1.2.

Таблица 1.2 – STEEP-анализ дальнего окружения компании «Инфиннити»

Фактор	Знак влияния	Качественная оценка	Вес	Бальная оценка	Взвешенная оценка	Критический синтез
Социальные факторы						
Возрастание требования потребителей к качеству приложений	-	Существенное	5	0,1	-0,5	Продвигать высокую квалификацию персонала.
Популяризация социальных сетей	+	Существенное	4	0,08	0,32	Внедрить в приложения элементы социальных сетей.
Технологические факторы						
Развитие прикладного, базового программного обеспечения для разработки информационных систем	+	Значительное	6	0,12	0,72	Использовать современные решения.
Развитие облачных технологий	+	Слабое	3	0,06	0,18	Продвигать себя как поставщика облачных решений.
Появление большого количества мобильных устройств с разными конфигурациями аппаратного и программного обеспечения	-	Значительное	7	0,18	-1,26	Усовершенствовать технологию тестирования мобильных приложений.

Окончание таблицы 1.2

Фактор	Знак влияния	Качественная оценка	Вес	Бальная оценка	Взвешенная оценка	Критический синтез
Экономические факторы						
Экономический кризис	-	Значительное	6	0,12	-0,72	Отслеживать экономическую ситуацию.
Возрастание доли финансовых вложений со стороны государства	+	Значительное	6	0,12	0,72	Принимать участие в государственных тендерах.
Политические факторы						
Федеральный закон №152 «О персональных данных»	-	Существенное	5	0,1	-0,5	Учитывать все требования закона.
Ухудшение отношений с Америкой	-	Слабое	2	0,04	-0,08	Налаживать отношения с неамериканскими компаниями.

Для оценки влияния факторов (Рисунок 1.2) были взяты данные из профиля состояния внешней среды: номер фактора и его важность.

Из данной диаграммы видно, что наиболее опасные для предприятия факторы под номерами 5 (технологический фактор «Появление большого количества мобильных устройств с разными конфигурациями аппаратного и программного обеспечения») и 6 (экономический фактор «Экономический кризис»). Из положительных сильно выделяется фактор под номерами 3 (технологический фактор «Развитие прикладного, базового программного обеспечения для разработки информационных систем») и 7 (экономический фактор «Возрастание доли финансовых вложений со стороны государства»).

Рисунок 1.2 – Профиль внешней среды

На основании проведенного анализа макросреда для компании «Инфиннити» является отрицательной. Сумма важности факторов равна -1,12, что свидетельствует о том, что окружающая среда является агрессивной, т.е. отрицательно воздействует на предприятие. Следовательно, компании «Инфиннити» необходимо повышать эффективность работы: увеличивать качество услуг, повышать скорость их выполнения. Это поможет сохранить положение «Инфиннити» на рынке.

1.4 Анализ ближнего окружения

Ближнее окружение – это элементы, с которыми непосредственно контактирует организация и она может оказывать свое влияние на эти элементы. Организации используют модель пяти конкурентных сил Портера для оценки микроокружения организации. Эти пять моделей представляют собой подробный анализ основных составляющих микросреды любой организации:

1. Потребители;
2. Партнеры, поставщики;
3. Конкуренты;

4. Угроза появления новых конкурентов;
5. Угроза появления товаров заменителей.

1. Рыночная власть потребителей

Потребителями продукции «Инфиннити» являются различные коммерческие, некоммерческие и государственные компании, нуждающиеся в специальных IT-решениях для их бизнеса. Также, к потребителям относятся пользователи сервисов Infinnity, например, пользователи сервиса «Классная.Москва». Ввиду того, что компания занимается разработкой программного обеспечения и IT-решений на заказ, каждый потребитель крайне важен для стабильной работы компании, и его потеря может привести к серьезным проблемам. Поэтому влияние потребителей серьезным образом сказывается на деятельности предприятия.

2. Рыночная власть поставщиков, партнеров

Так как у «Инфиннити» нет поставщиков, которые бы оказывали влияние на основной бизнес-процесс предприятия, то мы рассмотрим влияние партнеров.

Развитие продуктов компании происходит при поддержке зарубежных партнеров Ocean Informatics (Австралия), Microsoft (США), а также за счет использования открытого международного стандарта openEHR и самых передовых и инновационных технологий разработки программного обеспечения.

Взаимодействие с зарубежными партнерами в большей степени оказывает влияние на технологии, на основе которых и создаются информационные системы. Непосредственное общение с создателями стандарта openEHR, позволяет грамотно воспринимать имеющийся стандарт и в максимальной степени использовать его возможности. Сотрудничество с государственными органами позволяет создавать программный продукт, соответствующий всем требованиям законодательства.

Имеющиеся партнеры оказывают очень сильное влияние на основную деятельность организации. Именно поэтому политика компании направлена

не только на поддержание связей с существующими партнерами, но и на создание новой партнерской сети, которая позволила бы компании все больше прогрессировать.

3. Сила действующих конкурентов

Уже сейчас на рынке информационных систем прослеживается жесткая конкуренция. Связано это в первую очередь с тем, что данная отрасль является приоритетной в государстве, поддерживается и финансируется самим государством. Большее количество разработчиков сосредоточено в Москве. Но, тем не менее, регионы не отстают от центра.

Поскольку основными заказчиками систем у «Инфиннити» являются компании Москвы, в расчет берутся все компании-разработчики в РФ, без ограничения на одну Челябинскую область.

4. Угроза появления новых конкурентов

Появление новых конкурентов возможно, но появление действительно сильных конкурентов в краткий срок маловероятно, так как в отрасли высокая конкурентная борьба. Также новичкам будет довольно сложно преодолеть большие конкурентные отставания от лидеров, это может занять большое количество времени. К тому же игрокам нужно использовать прогрессивные технологии и выгодно отличаться от существующих конкурентов.

5. Угроза появления товаров-заменителей

На данном этапе развития информационных систем, товаров-заменителей у разрабатываемого продукта нет, угрозы их появления тоже.

Вывод по состоянию макросреды: отрасль разработки информационных систем является высоко-конкурентной. Однако конкурентные преимущества в виде инструмента для быстрой разработки собственных продуктов, большой опыт работы и высокая квалификация персонала позволяют компании не терять свои позиции.

1.5 Анализ внутренней среды предприятия

1.5.1 Организационная структура

Фирме «Инфиннити» присуща линейно-функциональная структура управления (рисунок 1.3). Такая структура обеспечивает выполнение специальных задач, контролируемых с помощью планов и бюджетов.

Так функциональные подразделения находятся в подчинении главного линейного руководителя. Свои решения они проводят через главного руководителя, либо (в пределах своих полномочий) непосредственно через соответствующих руководителей служб-исполнителей.

В «Инфиннити» имеется 4 основных отдела: администрация, единая служба сопровождения, отдел аналитики и департамент разработки. Департамент разработки разделен на три отдела: отдел разработки UI, отдел прикладной разработки и отдел тестирования.

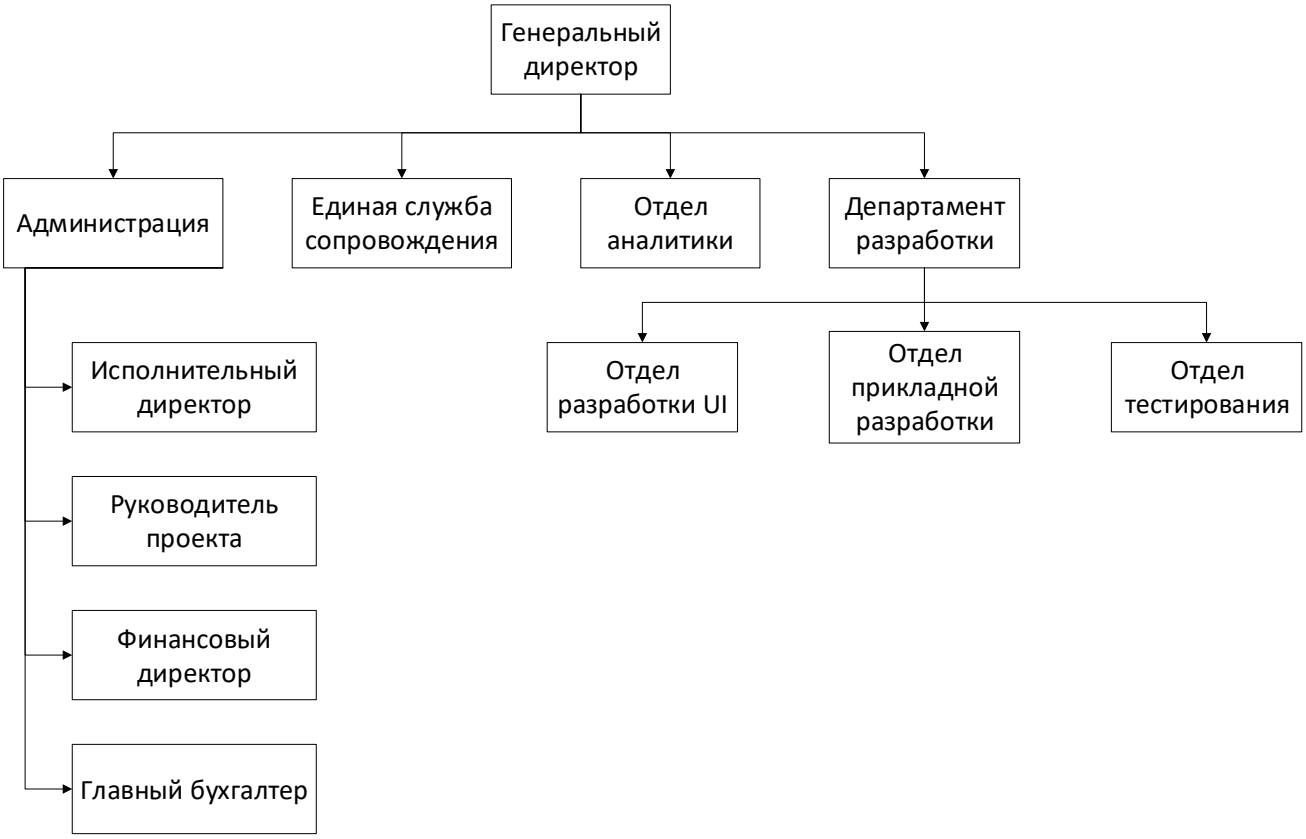

Рисунок 1.3 – Организационная структура компании «Инфиннити»

1.5.2 Выделение и идентификация бизнес-процессов

Бизнес-процесс — это совокупность взаимосвязанных мероприятий или задач, направленных на создание определённого продукта или услуги для потребителей [8].

Все бизнес-процессы делятся на основные, управляющие и обеспечивающие. К основным процессам компании «Инфиннити» относятся: разработка проектной документации, проектирование ИС, реализация ИС, тестирование ИС, внедрение ИС, сопровождение ИС. На рисунке 1.4 представлено дерево бизнес-процессов.

Процесс реализации ИС занимает центральное место в деятельности компании на сегодняшний момент, поскольку его результатом является готовый продукт, который должен удовлетворять потребностям заказчика. Данный процесс в нотации EPC показан на рисунке 1.5, а декомпозиция бизнес-процесса «Тестирование функционала» показана на рисунке 1.6.

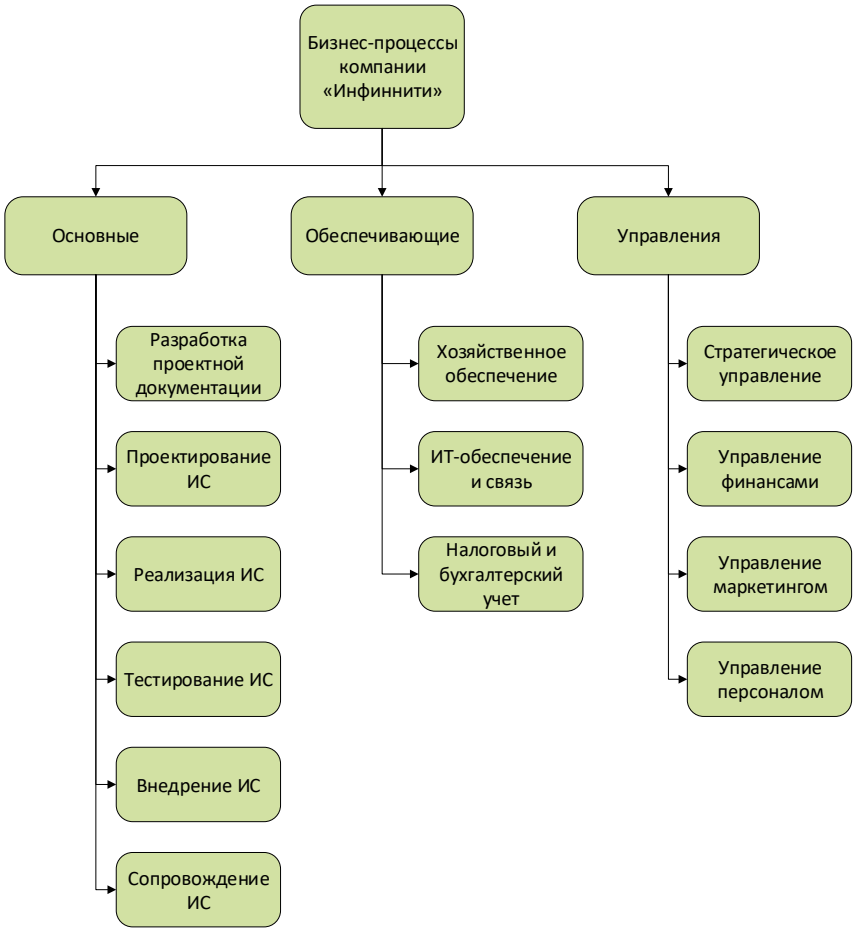

Рисунок 1.4 – Дерево бизнес-процессов компании «Инфиннити»

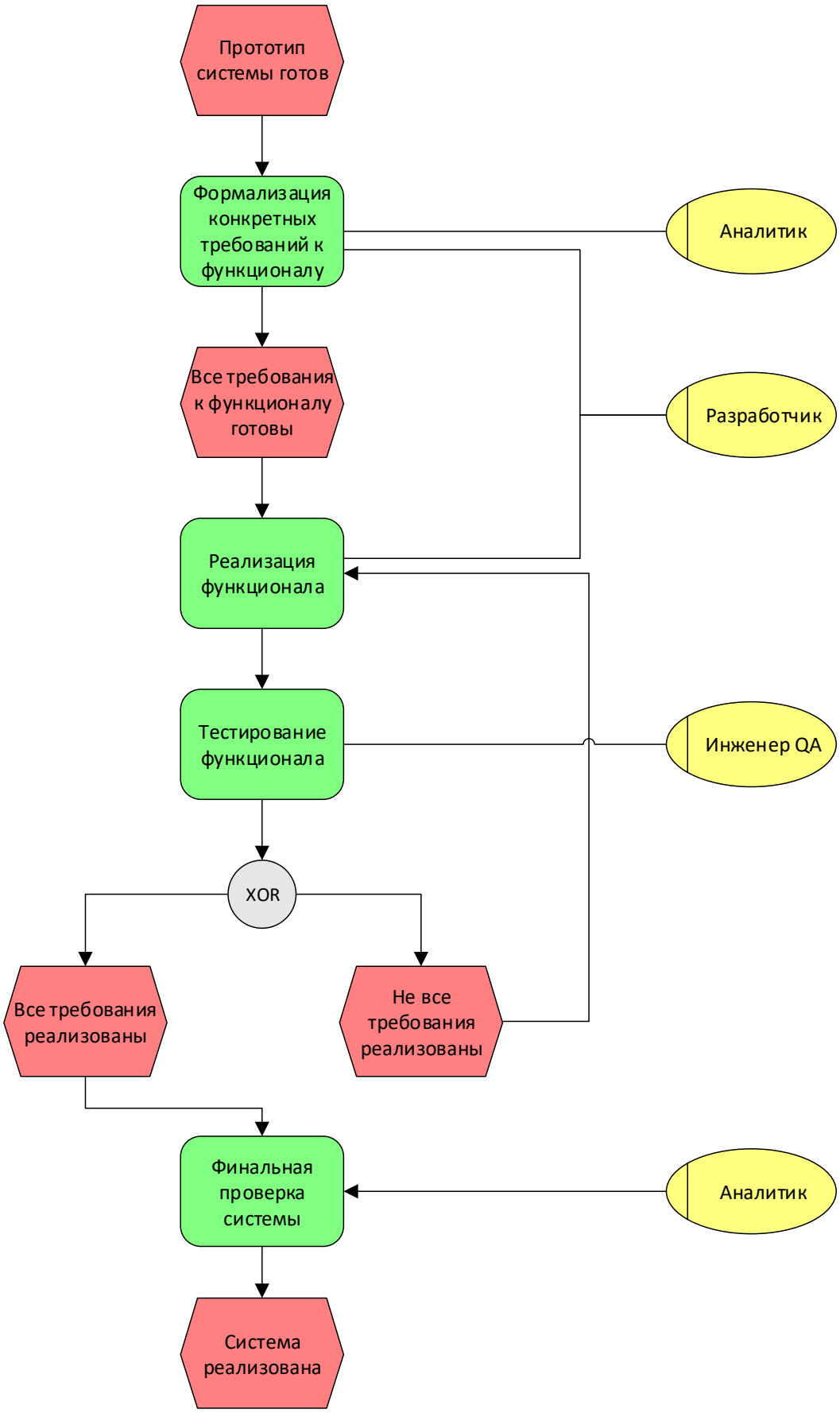

Рисунок 1.5 – EPC-диаграмма бизнес-процесса «Реализация ИС»

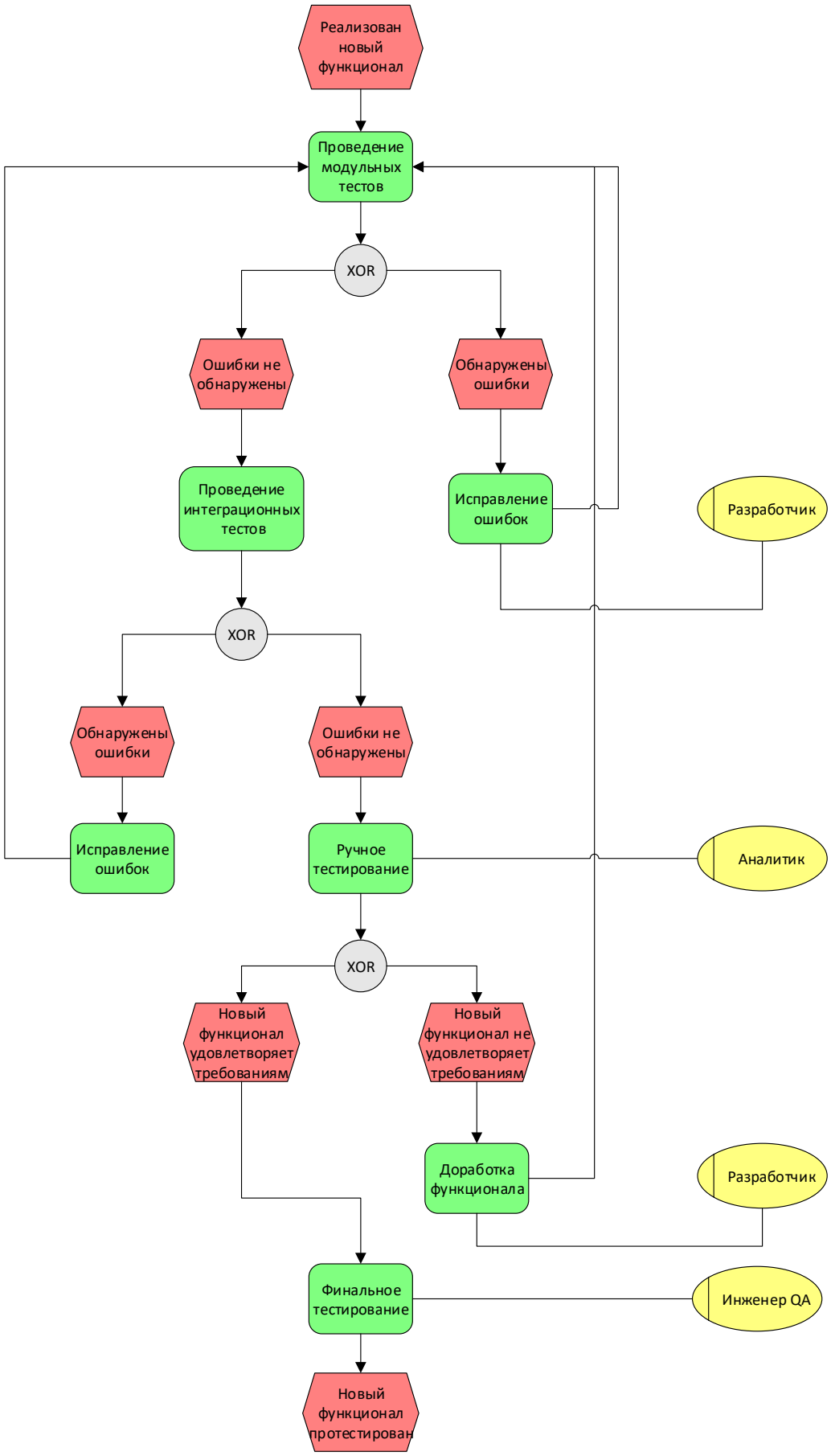

Рисунок 1.6 – Декомпозиция бизнес-процесса «Тестирование функционала»

На данной диаграмме видно, что последние этапы тестирования – это ручное тестирование. Поэтому, при большом количестве устройств, для которых разрабатывается система или приложение, данный этап занимает много времени. А в случае с мобильными приложениями отсутствует возможность протестировать это приложение на всех возможных комбинациях мобильных устройствах. Это приводит к появлению ошибок в приложении на определенных типах устройств.

1.5.3 Сравнительный анализ

Для определения сильных, слабых и нейтральных сторон компании был проведен SNW-анализ по категориям – кадры, управление, маркетинг, финансы, технологии (таблица 1.3). Так как конкурентов в области IT, предлагающих аналогичные услуги, большое количество и нет явных компаний конкурентов, то анализ был проведен в сравнении со среднестатистической компанией данной отрасли.

Таблица 1.3 – SNW-анализ

Внутренние факторы	S (сильные)			N (нейтральные)	W(слабые)		
	+3	+2	+1	0	-1	-2	-3
Кадры							
Квалификация персонала		●					
Мотивация			●				
Численность персонала						●	
Текучесть кадров			●				
Управление							
Управление персоналом			●				
Управление финансами				●			
Управление маркетингом				●			

Окончание таблицы 1.3

Технологии							
Следование тенденциям			●				
Компьютерное оборудование				●			
Собственные разработки		●					
Маркетинг							
Сбор необходимой информации о рынке				●			
Знание потребностей клиента			●				
Имидж, репутация и качество проектов			●				
Продвижение на рынке					●		
Финансы							
Отношение к налогам				●			
Финансовая устойчивость предприятия			●				
Заработная плата				●			

В результате проведенного SNW-анализа были выявлены сильны и слабые стороны компании «Инфиннити»

Сильные стороны:

- Низкая текучесть основного кадрового состава
- Высокая квалификация персонала
- Использование новых современных технологий в разработке ПО
- Продвижение собственных разработок
- Финансовая устойчивость предприятия

- Хорошая репутация, имидж и достойное качество проектов

Слабые:

- Малая численность персонала
- Слабое продвижение на рынке

1.6 Комплексный анализ

Каждая организация существует в определенной внешней среде, которая влияет и формирует ее структуру, создает положительные возможности и нежелательные обстоятельства, влияние которых различно, вследствие чего выделяют факторы микросреды (ближнее окружение) и макросреды (дальнее окружение). Для разработки устойчивой стратегии бизнеса предприятию необходимо проводить анализ этих факторов.

Для комплексного анализа используется методика SWOT-анализа.

SWOT-анализ компании «Инфиннити» представлен в таблице 1.4.

Таблица 1.4 – SWOT-анализ компании «Инфиннити»

Сильные стороны - S	Возможности - O
• Низкая текучесть основного кадрового состава • Высокая квалификация персонала • Использование новых современных технологий в разработке ПО • Продвижение собственных разработок • Финансовая устойчивость предприятия • Хорошая репутация, имидж и достойное качество проектов	• Развитие ПО для разработки ИС • Развитие облачных технологий • Популяризация социальных сетей • Возрастание доли финансовых вложений со стороны государства
Слабые стороны - W	Угрозы - T
• Малая численность персонала • Слабое продвижение на рынке	• Федеральный закон №152 «О персональных данных» • Экономический кризис • Появление различных конфигураций мобильных устройств

Результаты каждой стратегии:

SO – Возможности и сильные стороны

Сильные стороны предприятия будут способствовать реализации возможностей предприятия. Использование новых современных технологий в разработке ПО и высокая квалификация персонала позволят использовать все новые возможности, появляющиеся из-за развития ПО для разработки ИС, а также развития облачных технологий. Также хорошая репутация, имидж и достойное качество проектов помогут компании получить заказы от государства и бюджетных организаций.

OW - Возможности и слабые стороны

Слабые стороны предприятия могут негативно сказаться на реализации возможностей, диктуемых внешней средой. Малая численность персонала может негативно сказаться на использовании возможностей, появляющихся из-за развития ПО для разработки ИС. Кроме того, слабое продвижение на рынке может помешать получению финансирования от государства.

TS – Угрозы и сильные стороны

Сильные стороны предприятия могут минимизировать или полностью нейтрализовать возможные угрозы. Использование современных технологий в разработке ПО позволит нивелировать появление различных конфигураций мобильных устройств. А финансовая устойчивость предприятия позволит удовлетворить все требования федерального закона №152 «О персональных данных».

WT – Слабые стороны и угрозы

Соотношение слабых сторон предприятия и угроз, продиктованных внешней средой. Угроза экономического кризиса вместе с слабым продвижением на рынке может привести к потере клиентов. Также малая численность персонала вместе с появлением различных конфигураций мобильных устройств может привести к проблемам в разработке мобильных приложений, поскольку тестирование таких приложений сильно усложняется.

Вывод: проанализировав матрицу SWOT-анализа, была выбрана стратегия «SO». Компании «Инфиннити» необходимо использовать свои сильные стороны для получения отдачи от возможностей. Для этого необходимо продолжить использовать новые современные технологии для разработки ПО, а также большое внимание уделить получению заказов от государства.

1.7 Классификация и ранжирование проблем предприятия

Для определения наиболее важных из существующих проблем для дальнейшей разработки их решения была построена матрица Глайстера (таблица 1.5).

Таблица 1.5 – Матрица Глайстера

Уровни	Суть проблемы	Признаки проявления	Рекомендации
Отдел тестирования	Тестирование мобильных приложений проводится на недостаточном количестве устройств	Ошибки в работе мобильных приложений обнаруживаются и исправляются уже после выхода приложения на рынок	Внедрение модуля автоматизированного тестирования мобильного приложения на реальных устройствах.
	Малая производительность тестирования мобильных приложений	Длительное время выполнения приемочного тестирования	

Мобильные приложения в «Инфиннити» разрабатываются одновременно для операционных систем iOS и Android. И на мобильных устройствах могут быть установлены разные версии данных систем. Эти версии могут расширять функционал системы или изменять существующий. К тому же сами мобильные устройства бывают с различными характеристиками. Это касается и размера оперативной памяти, и мощности процессора, и многих других технических характеристик.

Отсюда возникает проблема: приложение, которое хорошо работает на каком-то конкретном устройстве, может плохо работать на другом. Например, могут появиться проблемы с отображением интерфейса: другие цвета,

неожиданное появление фона (рисунок 1.7). Поэтому возникает необходимость протестировать приложение на разных устройствах с разными версиями операционных систем и с разными техническими характеристиками.

На данный момент в «Инфиннити» проводится ручное тестирование мобильных приложений на небольшом комплекте реальных устройств. Но зачастую этого количества устройств бывает недостаточно, и после выхода приложения на рынок, могут проявиться новые проблемы.

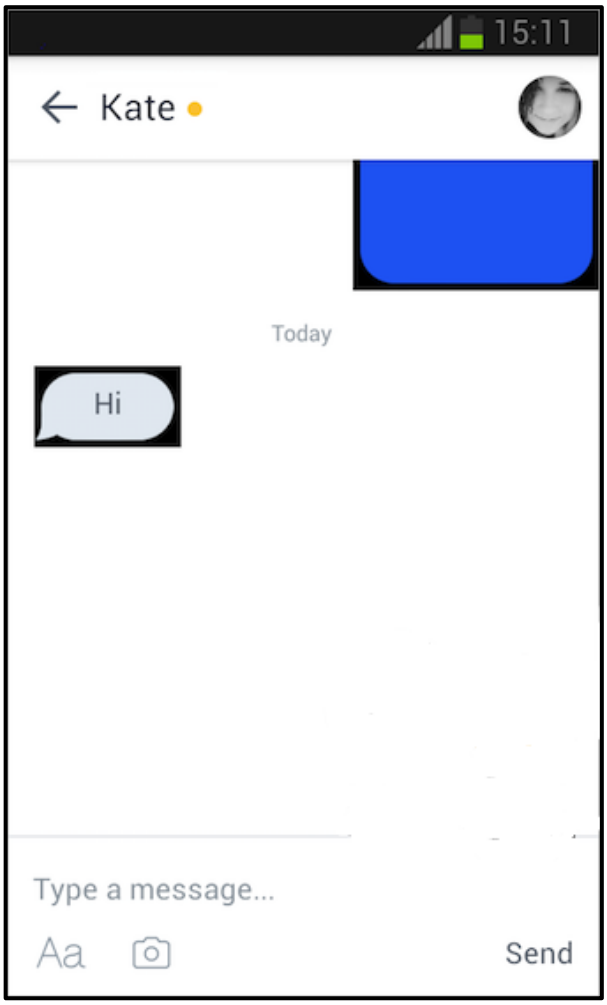

Рисунок 1.7 – Пример некорректного отображения интерфейса

Выводы по главе 1

В первой главе были проанализированы ближнее и дальнее окружение фирмы, выявлены сильные и слабые стороны организации, а также были выявлены проблемы, связанные с тестированием мобильных приложений.

Что касается внешней среды, то для «Инфиннити» макросреда является отрицательной. Внешняя среда оказывает весьма сильное влияние на деятельность организации, поэтому обеспечение предприятия полностью рабочей и хорошей ИС является очень важным фактором.

И чтобы противостоять внешней среде, компании «Инфиннити» необходимо повышать качество разрабатываемых продуктов, уменьшать время разработки. Поэтому необходимо избавляться от проблем, которые негативно влияют на качество итогового продукта и на скорость его разработки. В отделе тестирования эта проблема необходимостью тестирования мобильных приложений на как можно большем количестве устройств.

ГЛАВА 2. РАЗРАБОТКА ИНФОРМАЦИОННОГО МОДУЛЯ ДЛЯ РЕШЕНИЯ ПРОБЛЕМ ПРЕДПРИЯТИЯ

2.1 Словарь терминов

Некоторые из упоминаемых терминов предметной области являются специализированными и требуют пояснений. Ниже дано описание основных понятий, используемых в описании предметной области.

Логи (лог-файлы) – это файлы, содержащие системную информацию работы сервера или компьютера, в которые заносятся определенные действия пользователя или программы. Их предназначение - протоколирование операций, выполняемых на машине, для дальнейшего анализа. Просмотр логов позволит определить ошибки в работе системы в целом, конкретного сервиса или приложения.

Эмулятор – комплекс программных средств, который полностью или частично предоставляет функционал и поведение устройства или программы. При разработке мобильных приложений используются эмуляторы мобильных устройств, которые позволяют оперативно тестировать приложения, даже когда целевое устройство недоступно.

Облако (облачные вычисления) – это технология распределённой обработки данных в которой компьютерные ресурсы и мощности предоставляются пользователю как интернет-сервис.

Инженер QA – должность в компании «Инфиннити», ответственная за проведение тестирования приложений и информационных систем.

Непрерывная интеграция (от англ. Continuous Integration) – это практика разработки программного обеспечения, в которой при добавлении или изменении кода приложения сборка автоматически компилируется и, при необходимости, тестируется.

Фреймворк – это программный продукт, позволяющий упростить и ускорить решение типовых задач. Фрейм-ворк, как правило, содержит только

базовые программные модули, а все специфичные для проекта компоненты реализуются разработчиком на их основе.

2.2 Описание интернет-сервиса «Классная Москва»

В настоящее время компания «Инфиннити» разрабатывает проект «Классная Москва». Цель проекта – создать интернет-сервис для родителей школьников, с помощью которого пользователи могли бы планировать и организовывать внеклассные мероприятия [4].

На текущем этапе разработки для проекта «Классная Москва» характерны следующие особенности:

1) Пользователями интернет-сервиса являются родители школьников;

2) Пользователи могут:

- Регистрировать классы, приглашать в них других родителей;
- Создавать события;
- Создавать и участвовать в голосованиях;
- Организовывать сборы денег;
- Просматривать каталог идей;
- Вести чат с другими родителями;
- Просматривать календарь событий.

3) Интернет-сервис существует в виде web-сайта и мобильного приложения в двух версиях: для устройств на базе Android и iOS.

4) Сервис «Классная Москва» находится в фазе открытого тестирования и доступен для пользования.

Разработка мобильной версии данного приложения ведется с помощью фреймворка Xamarin. Xamarin – это фреймворк для кроссплатформенной разработки мобильных приложений (iOS, Android, Windows Phone) с использованием языка C# [3].

Алгоритмы, разработанные на C# для одной платформы, программисты используют и для всех остальных платформ. Например, при работе на

Xamarin.Android и Xamarin.iOS, до 75% кода можно переиспользовать: приложение работает одинаково и на iPhone, и на Android-смартфоне.

Но кроссплатформенность имеет и свои недостатки, один из которых – тестировать такие приложения гораздо сложнее и затратнее.

2.3 Общие вопросы тестирования мобильных приложений

Тестирование – крайне важный этап разработки мобильных приложений. Мобильный пользователь ожидает, что устанавливаемые им приложения просты, интуитивно понятны, работают без сбоев. Если ожидания не оправдываются, то пользователь может отказаться от использования приложения.

При этом в тестировании мобильных приложений есть очевидная трудность – огромное количество вариаций мобильных устройств. Это происходит из-за таких характеристик устройства, как операционная система (Android, iOS, Windows Phone), версия операционной системы, разрешение и размеры экрана, емкость батарейки, оператор, количество сим карт, наличие или отсутствие WiFi и многих других.

Поэтому возникает ряд задач, которые необходимо решить для качественного тестирования:

- Установка приложения – прежде всего для тестирования необходимо установить приложение.
- Сбор логов – во время выполнения приложения должны собираться лог-файлы. В дальнейшем их необходимо собирать и обрабатывать.
- Снятие снимков экрана и видео – для оценки корректной работы приложения необходимо собирать как статичные снимки экрана, так и видео (например, для проверки анимаций).
- Оценка интерфейса – необходимо оценивать, насколько текущий интерфейс приложения соответствует требованиям к приложению.

- Сбор показателей производительности – необходимо замерять показатели производительности во время работы приложения, а также собирать эти результаты. Среди таких показателей – объем потребляемой оперативной памяти, нагрузка на процессор, потребление батареи.
- Эмуляция различных прерываний работы приложения – во время реального использования приложения могут возникнуть различные прерывания: звонки, получение SMS, отключения устройства из-за батареи. Поэтому и при тестировании необходимо обеспечить эмуляцию таких прерываний.

Выбор инструментов, которые могут использоваться для этих задач, зависит от конкретного приложения, от требований к нему и от его функционала.

Существует четыре основных способа решения данных задач. Это тестирование на реальных устройствах, тестирование на эмуляторах, тестирование на реальных устройствах через облако и тестирование на эмуляторах через облако.

Тестирование на реальных устройствах

Тестирование на реальных устройствах – довольно простой и очевидный способ тестирования. В настоящее время найти Android или iOS-устройство не составляет большого труда.

Такое тестирование может показать наиболее реалистичный опыт взаимодействия с приложением. Ведь именно на реальных устройствах пользователи будут использовать приложение.

Но основная проблема здесь – огромное количество самих устройств. Как итог – для полноценного проведения теста необходимо их большое количество. Но даже если приобрести определенное количество устройств с различными вариантами программного и аппаратного обеспечения, этого количества все равно может не хватить. И на каких-то не очень популярных устройствах могут обнаружиться ошибки в работе приложения.

Тестирование на эмуляторах

Основное преимущество эмуляторов – они дают мгновенный доступ к любому устройству или версии, доступной в настоящее время. Соответственно, эмуляторы позволяют тестировать приложения на самых различных конфигурациях устройств. Это экономит и время, и место, и деньги.

Но ключевой недостаток здесь - на эмуляторах нет возможности узнать, как приложение действительно выглядит в реальности. Например, сейчас широкое распространение набирают AMOLED-дисплеи – новая технология создания дисплеев. И конечное изображение, которое увидит пользователь, вполне возможно будет отличаться от того, которое тестировщики увидят на эмуляторе.

Также эмуляторы не позволяют использовать различные аппаратные функции устройства, например, датчик отпечатков пальцев.

Тестирование на реальных устройствах через облако

Облачные технологии дают преимущества реальных устройств, не владея ими. Этот способ реализуется следующим образом: компания покупает сотни устройств, но вместо того, чтобы хранить их для себя, она делится ими через технологии облачных вычислений. Так, например, это делает компания Amazon.

Эта услуга не бесплатная, но взамен можно сэкономить как время, так и место.

Основная проблема здесь – производительность. Соединение с устройствами посредством облачных технологий не всегда стабильно. Кроме того, порой некоторые устройства непригодны для использования. Это происходит потому, что нельзя запускать теста на уже используемых устройствах. Поэтому иногда перед тем, как получить доступ к нужным телефонам, может возникнуть огромная очередь.

Тестирование на эмуляторах через облако

Данный способ представляет из себя что-то среднее между тестированием на эмуляторах и тестированием на реальных устройствах с помощью облака. За основу здесь берется облачная технология, но вместо реальных устройств устанавливаются эмуляторы.

Основное преимущество такого способа - совместное проведение тестирования. В том числе в ситуациях, когда тестировщики находятся на удалении друг от друга.

Преимущества и недостатки каждого из способов тестирования приведены в таблице 2.1.

Таблица 2.1 – Сравнение способов тестирования

Способ тестирования	Преимущества	Недостатки
Тестирование на реальных устройствах	• Наиболее реальные условия работы с приложением • Лучшая производительность	• Слишком большое количество вариаций устройств и Android-оболочек • Под устройства требуется место и оборудование • Удаленные команды не могут получить доступ к устройствам
Тестирование на эмуляторах	• Доступ ко всем версиям Android и большинству устройств • Простота настройки • Большое количество устройств можно запускать сразу, не требуя дополнительного физического пространства • Относительная дешевизна	• Некоторые аппаратные функции не могут быть протестированы • Эмуляторы не всегда позволяют увидеть, как приложение выглядит в реальности • Не предназначены для коллективной работы
Тестирование на реальных устройствах через облако	• Доступ к большому количеству физических устройств без непосредственного владения ими • Не требуется дополнительное место или обслуживание • Подробные отчеты после каждого теста	• По определению физический доступ к устройствам невозможен • Низкая производительность и стабильность • Запрошенные устройства могут занять много времени до того, как они будут доступны • Различные ограничения, связанные с конкретным сервисом

Окончание таблицы 2.1

Тестирование на эмуляторах через облако	• Доступ к эмуляторам из любого места • Доступ ко всем версиям Android и большинству устройств • Не требуется дополнительное место или обслуживание	• Относительная дороговизна услуги • По-прежнему отсутствует возможность протестировать определенные аппаратные функции

Вывод: выбор конкретного способа тестирования зависит от целей и задач создания приложения, а также этапа тестирования. Например, на ранних этапах разработки приложения применяется тестирование с помощью эмуляторов. Это позволяет в короткие сроки посмотреть работу приложения, выявить критические ошибки. А при выходе на массовый рынок очень важно провести тестирование на реальных устройствах. При отсутствии большого количества реальных устройств можно провести тестирование через облако.

Поэтому на практике используются различные сочетания данных способов тестирования. Так в компании Инфиннити во время разработки приложения используется тестирование на эмуляторах, а в дальнейшем проводится тестирование на комплекте из нескольких реальных устройств.

2.4 Описание бизнес-процесса тестирования «as is»

Тестирование мобильного приложения в «Инфиннити» начинается уже на этапе разработки. После создания нового функционала разработчики сразу же закладывают в него юнит-тесты и UI тесты.

Юнит-тестирование (англ. unit testing), или модульное тестирование – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы. Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или процедуры. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Модульный тест применяется для тестирования одной логически выделенной и изолированной единицы системы. Чаще всего это метод класса или простая функция [6].

Так, сервис «Классная Москва» состоит из определенных модулей. Это модули: события, календарь, голосования, объявления, поручения и другие. Программная реализация каждого модуля состоит из набора функций и процедур. И для каждой такой функции и процедуры написаны юнит-тесты, которые запускаются автоматически каждый раз, когда реализована новая часть функционала.

Например, для модуля события существуют функция создания события. Для события указываются определенные атрибуты: название события, его описание, место встречи, дата проведения, время проведения, координатор. У каждого из атрибутов есть определенный тип данных: для названия, описания, места встречи – текстовый; для даты и времени проведения – тип дата; для координатора – специальный тип «родитель». И один из юнит-тестов, который будет проводиться при тестировании – это соответствие получаемых типов данных заложенным в систему. Например, если при создании события значение времени проведения станет текстовым вместо типа дата, будет выдана ошибка.

Пример юнит-теста на языке JS приведен на рисунке 2.1.

```
describe("CreateEvent", function() {

    it("Title must be string type", function() {
      assert.equal(typeof(context.title), "string");
    });

    it("Description must be string type", function() {
      assert.equal(typeof(context.description), "string");
    });

});
```

Рисунок 2.1 – Пример юнит-теста на языке JS

UI-тесты (англ. UI – user interface), или тестирование интерфейса пользователя – тестирования уже самого интерфейса приложения, всех его функций. Заключается в том, что мы имитируем действия пользователя – клики, переходы по ссылкам, и другие действия подобного плана. Смысл его – в проверке взаимодействия модулей приложения друг с другом.

При создании UI-тестов используется синтаксис предметно-ориентированного бизнес-понимаемого языка Gherkin. Синтаксис языка укладывается в шаблон, описывающий начальные условия, действия пользователя и конечный ожидаемый результат. Пример сценария на языке Gherkin:

«Функционал: Создание нового события.

Scenario: Проверка функционирования команды «Создать событие» в меню «События»

Given: Пользователем открыто меню «События»

When: Нажать на кнопку «Создать событие»

Then: Открывается форма создания нового события.

When: Ввести данные, нажать «Создать»

Then: Появляется оповещение об успешном создании события.»

На основании данных сценариев генерируются тесты. Тесты создаются в декларативном стиле, с помощью «шагов». Их генерацией занимается фреймворк SpecFlow. SpecFlow позволяет записать тестовые сценарии в Given When Then стиле, а затем автоматизировать их. Запускать тесты на выполнение можно как из среды разработки, так и с помощью используемой на предприятии программы TeamCity.

TeamCity – это многофункциональный сервер непрерывной интеграции, готовый к работе сразу же после установки. Он поддерживает множество систем контроля версий, аутентификации, сборки и тестирования прямо из коробки. При этом TeamCity легко расширяем.

TeamCity обеспечивает интеграцию и контроль модификаций кода в процессе совместной работы и автоматизирует процесс запуск тестов. Решение контролирует комплексную работу тестов и исправление ошибок и осуществляет генерацию оповещений. Благодаря мощной функциональности TeamCity позволяет эффективно управлять разработкой и предоставляет возможность использования всех возможных типов конфигурации для любых проектов.

Ключевые возможности TeamCity:

- Функция Build Chains позволяет разделять процедуру сборки на части и выполнять обработку фрагментов кода на нескольких агентах последовательно или параллельно.
- Возможность изменения порядка прохождения тестов и запуска наиболее сложных тестов первыми при изготовлении очередной сборки.
- Встроенные механизмы контроля версий позволяют воссоздать необходимую версию проекта на любом этапе процесса разработки.
- Расширенные возможности аутентификации агентов.
- Возможность просмотра подробной статистической информации по проекту в целом и сравнительной статистики по каждой версии проекта.
- Более тесная интеграция с платформой Eclipse предоставляет возможность воспользоваться многочисленными функциями и инструментами.
- После регистрации пользовательских AMI (изображений сервера Amazon), входящих в Build Agent, сервер TeamCity учитывает эти виртуальные машины в своем расписании и автоматически применяет к ним команды приостановки/возобновления, основываясь на их размере и порядке очереди.
- Шаблоны конфигурации разработки

- Пользователь может создать шаблон со стандартными (общими) настройками, а затем скопировать из этого шаблона любое количество конфигураций разработок.
- Интеграция инструмента отслеживания событий
- Средство командной строки для персональных разработок
- Пользователь может осуществлять разработку локальных изменений удаленно с сервера с помощью командной строки.

Проведение модульных и UI тестов позволяет в кратчайшие сроки находить и исправлять ошибки. Проводятся они в автоматическом режиме и при возникновении неполадок позволяют точно определить, из-за какого модуля программа перестала работать корректно.

После проведения этих тестов, разработчик передает приложение аналитику. Тот проверяет работу приложение вручную путем обхода функционала раз за разом, и сверяет поведение системы с тем, что прописано в техническом задании. Качество такого тестирования далеко неидеальное – человек может упустить что-то из виду, забыть или полениться. Более того на такое тестирование уходит большое количество времени. Один из главных недостатков такого тестирования – момент времени обнаружения ошибки в поведении системы и ее исправления очень удалены друг от друга. Иногда проходит несколько дней, неделя или даже больше времени на ее выявление, на поиск причины ее возникновения и устранения может уйти еще неопределенное количество времени. Такое, не всегда зависящее от человеческого фактора, промедление негативно отражается на сроках сдачи, оговоренных с заказчиком. Зачастую процесс тестирования занимает на порядок больше времени, чем процесс составления технического задания на разработку.

Следующим шагом приложение от аналитика передается инженеру QA (от англ. quality assurance – обеспечение качества). Здесь приложение проходит последний этап тестирования.

Инженер QA вручную исследует приложение с целью поиска нестандартных сценариев работы и ошибок, связанных с ними. После успешного тестирования данная версия становится текущей стабильной версией приложения.

2.5 Описание внедряемого решения

В настоящее время многими компаний при разработке приложений активно используются методы непрерывной интеграции.

Обычно в программных проектах разработчики работают параллельно. В какой-то момент необходимо объединить все эти параллельные потоки работы в одну кодовую базу, которая составляет конечный продукт.

Непрерывная интеграция позволяет избегать сложностей путем слияния изменений каждого разработчика с общей базой кода на постоянной основе. Каждая регистрация запускает автоматическую сборку и запускает автоматические тесты, чтобы убедиться, что новый код не нарушил существующий код. Таким образом, непрерывная интеграция немедленно позволяет находить ошибки и проблемы.

Системы непрерывной интеграции имеют две основные части:

- Управление версиями – управление версиями объединяет весь код проекта в один общий репозиторий и сохраняет всю историю каждого изменения для каждого файла. Этот репозиторий, часто называемый основной или ведущей ветвью, содержит исходный код, который в конечном итоге будет использоваться для создания производственной или выпускной версии приложения. Для этой задачи существует множество продуктов с открытым исходным кодом и коммерческих продуктов, которые обычно позволяют командам или отдельным лицам разворачивать копию кода во вторичные отрасли, где они могут вносить значительные изменения или проводить эксперименты без риска для основной ветви. После

того, как изменения во вторичной ветви будут проверены, они могут быть объединены вместе в главную ветвь.

- Сервер непрерывной интеграции – сервер непрерывной интеграции отвечает за сбор всех артефактов проекта (исходный код, изображения, видео, базы данных, автоматические тесты и т.д.), компиляцию приложения и запуск автоматических тестов.

Разработчики обычно имеют рабочую копию одной или нескольких веток. После того, как соответствующий набор работ будет завершен, изменения будут «проверены» или «переданы» соответствующей ветке, которая распространяет их на рабочие копии других разработчиков. Именно так команда гарантирует, что все они работают над одним и тем же кодом.

Опять же, при непрерывной интеграции акт совершения изменений заставляет сервер непрерывной интеграции строить проект и запускать автоматические тесты для проверки правильности исходного кода. Если возникают ошибки сборки или сбои в тестировании, сервер непрерывной интеграции уведомляет ответственного разработчика, чтобы он мог исправить проблему.

Рисунок 2.2 иллюстрирует этот процесс.

При использовании данного решения для разработки приложений для мобильных устройств отличным решением будет использовать тестирование на реальных устройствах через облако. При внедрении данного способа сервер непрерывной интеграции будет автоматически отправлять новую версию приложения для тестирования. Это позволит на ранних этапах находить ошибки и проблемы, связанные с конечным отображением приложения на мобильном устройстве. При этом такое тестирование станет частью непрерывной интеграции, как показано на рисунке 2.3.

Процесс «Тестирования» «to be» показан на рисунке 2.4.

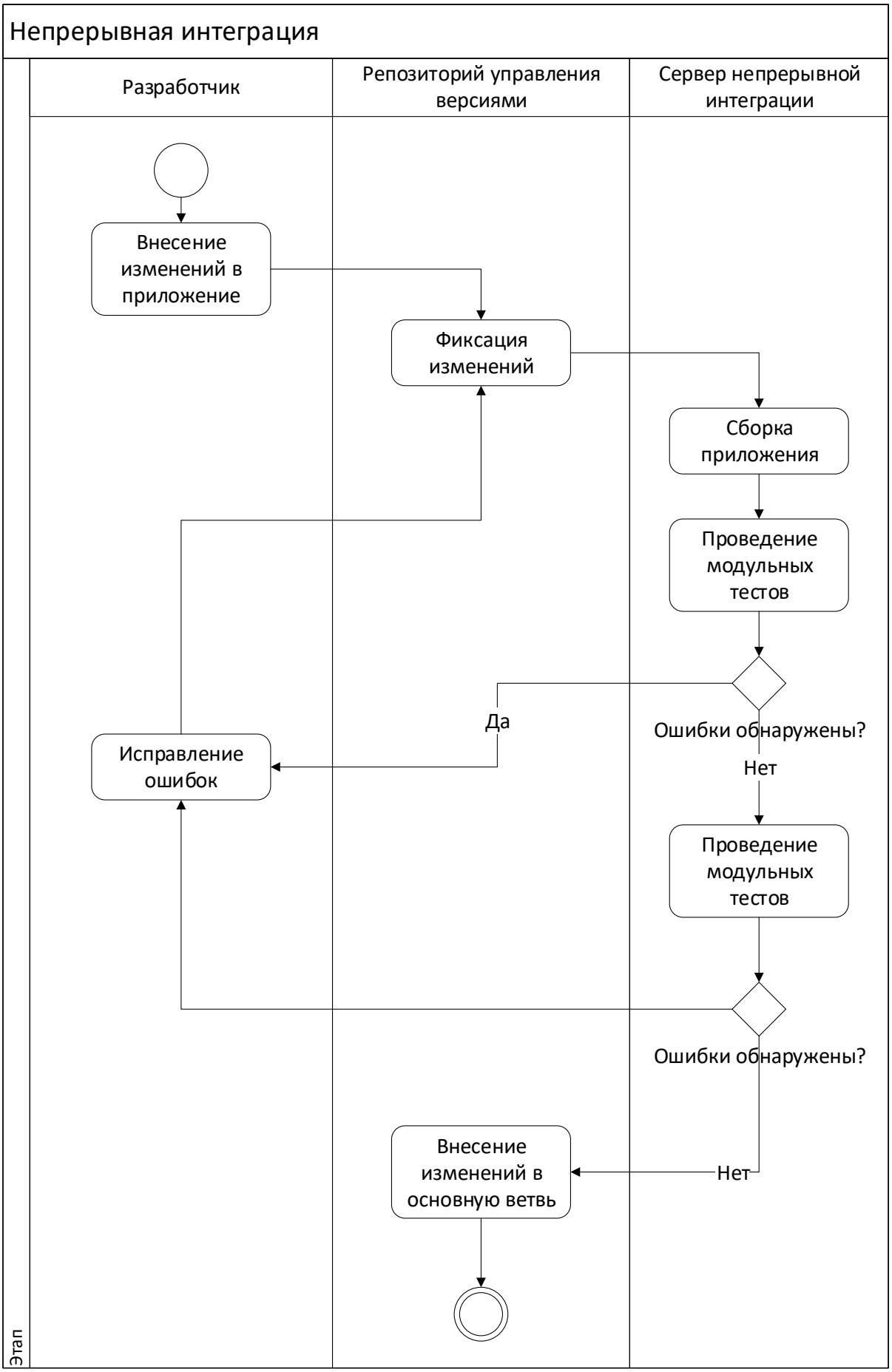

Рисунок 2.2 – Процесс разработки приложения «as is»

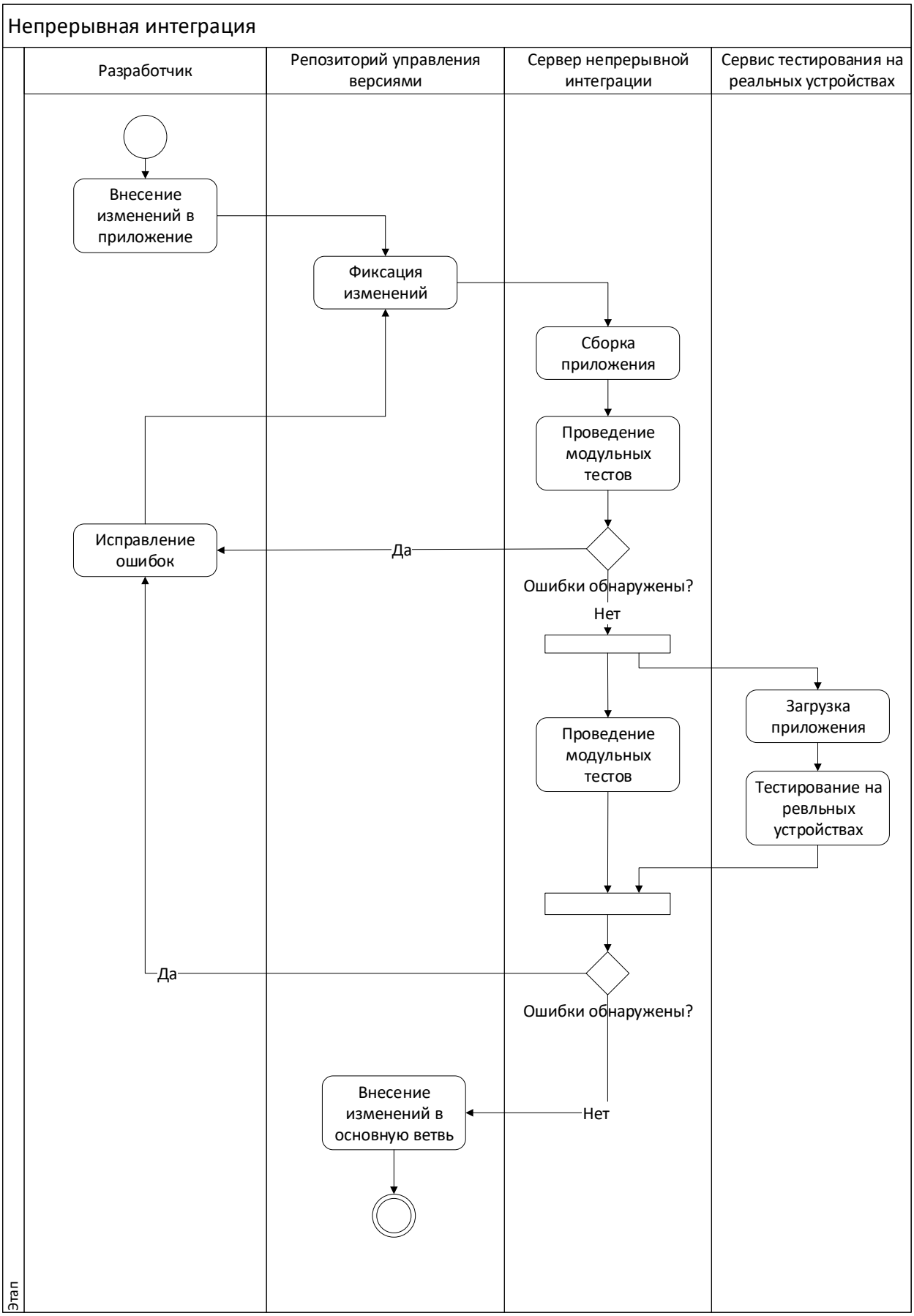

Рисунок 2.3 – Процесс разработки приложения «to be»

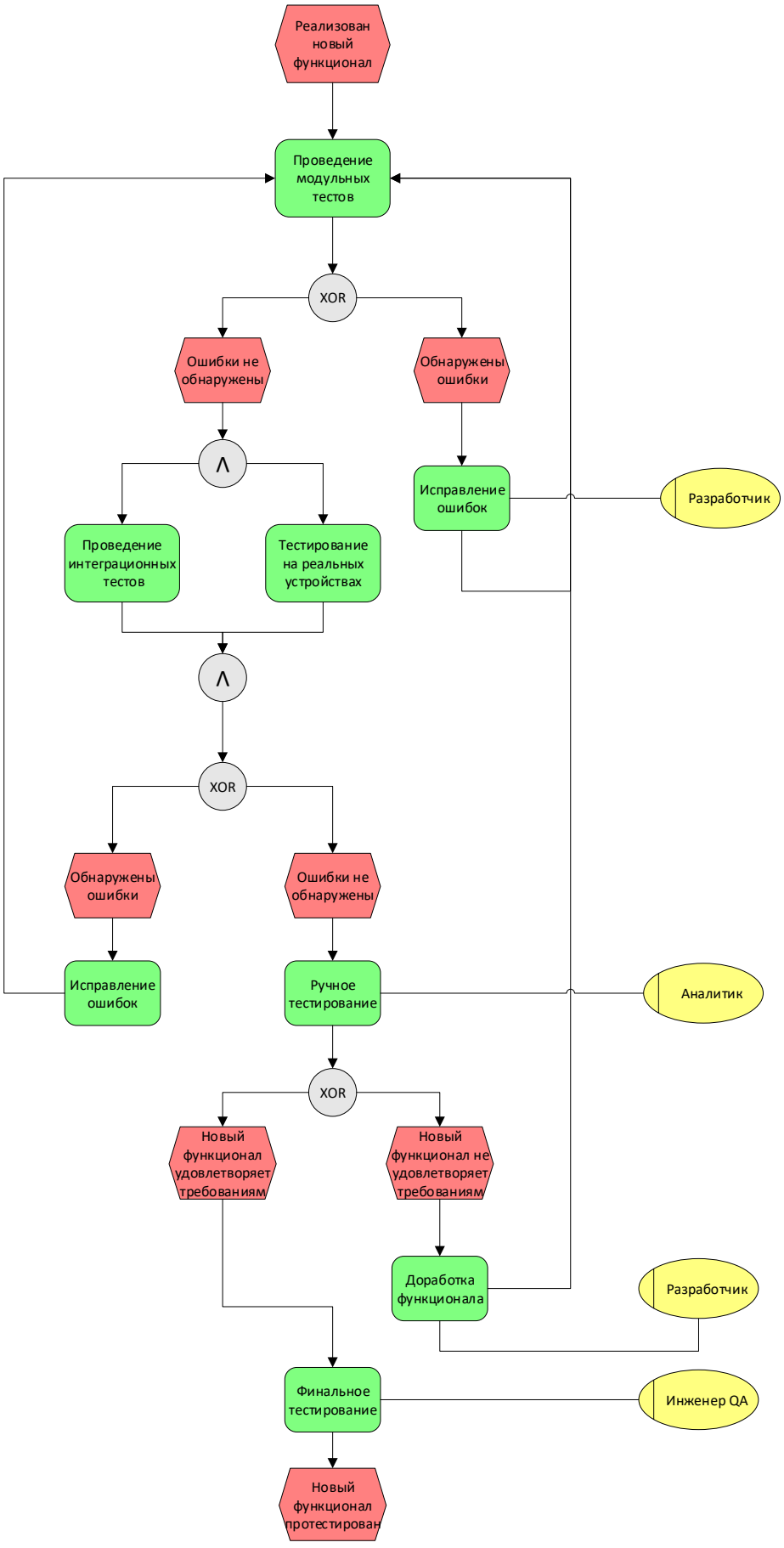

Рисунок 2.4 – Процесс «Тестирование» «to be»

2.6 Сравнение сервисов тестирования мобильных приложений

Растущая популярность облачного тестирования привела к появлению большого количества новых инструментов тестирования на рынке.

Основные сервисы:

1. App Center Test (Xamarin Test Cloud)

App Center Test (также известный как Xamarin Test Cloud) – это сервис автоматизации тестирования для мобильных приложений. Тесты, написанные с использованием поддерживаемых фреймворков, могут выполняться с небольшими изменениями на сотнях уникальных моделей устройств и конфигураций операционной системы, размещенных в центре обработки данных Microsoft. App Center сохраняет результаты тестирования, включая все связанные с ним медиа-файлы, для просмотра в любое время[5].

1. Perfecto Mobile Continuous Quality Lab

Perfecto Mobile Continuous Quality Lab — специальный набор инструментов для тестирования и оптимизации работы мобильных приложений. Помимо тестирования стандартных функций устройств данный сервис дает возможности совершать звонки, отсылать смс, использовать Internet. Это возможно благодаря подключению устройств к основным мобильным операторам США, Великобритании, Индии, Канады, Израиля и других стран.

2. SIGOS App Experience

App Experience предоставляет облачные решение для соединения между собой устройств проводной связи. Все операции осуществляются с помощью фреймворка Keynote, для чего необходима установка соответствующего программного обеспечения и его постоянное обслуживание.

Преимущества и недостатки данных сервисов приведены в таблице 2.2.

Таблица 2.2 – Сравнение сервисов тестирования приложения на реальных устройствах

Сервис	Преимущества	Недостатки
App Center Test	1. Большое количество устройств и конфигураций для тестирования 2. Тестирование на различных ОС (iOS, Android, Windows Mobile и Blackberry) 3. Тщательное тестирование производительности 4. Удобные отчеты 5. Невысокая цена	1. Не поддерживаются определенные функции устройств (Bluetooth, WiFi, камера) 2. Временные ограничения для определенных тестов
Perfecto Mobile Continuous Quality Lab	1. Есть возможность протестировать звонки, СМС, Internet 2. Одновременное тестирование на нескольких устройствах 3. Возможность интеграции с большинством сред разработки	1. Высокая цена 2. Неудобные отчеты о тестировании 3. Нет возможности ручного тестирования
App Experience	1. Параллельное исполнение нескольких тестов на нескольких устройствах 2. Настройка отчетов в соответствии с потребностями 3. Возможность ручного тестирования	1. Отсутствует поддержка резервирования устройств 2. Высокая цена

2.7 Выбор сервиса для внедрения

Выбор наиболее подходящего продукта осуществляется по критериям, приведённым в таблице 2.3. Каждый критерий имеет свой установленный

весовой коэффициент. Каждому из критериев была проставлена оценка экспертным путем (максимальная оценка равна 5). Взвешенная оценка вычисляется произведением значений веса и оценки.

Таблица 2.3 – Критерии выбора сервиса

Критерий	Вес	App Center Test		Perfecto Mobile Continuous Quality Lab		App Experience	
		Оценка	Вз. оценка	Оценка	Вз. оценка	Оценка	Вз. оценка
Функциональная полнота	0,25	5	1,25	4	1	5	1,25
Гибкость конфигурации	0,12	4	0,48	4	0,48	4	0,48
Целевая определенность	0,15	4	0,6	3	0,45	4	0,6
Простота использования	0,13	4	0,52	5	0,65	5	0,65
Возможность интеграции	0,2	5	1	4	0,8	3	0,6
Цена	0,15	5	0,75	3	0,45	3	0,45
	1		4,6		3,83		4,03

В диапазон 4,5 - 5 (подходящие ИС) попал сервис App Center Test. Но помимо данных критериев внедрение именно этого сервиса хорошо и по другой причине: в компании «Инфиннити» разработка приложений ведется в среде Xamarin. И сервис App Center Test предоставляется той же фирмой, поэтому Xamarin и App Center Test отлично интегрируются друг с другом.

Исходя из вышеперечисленного, наиболее оптимальным выбором будет внедрение тестирование приложения на реальных устройствах с помощью сервиса App Center Test.

2.8 Внедрение сервиса

Для внедрения сервиса App Center Test для приложения «Классная.Москва» и проведения тестов на реальных устройствах необходимо проделать следующие шаги [1]:

1) На сайте https://github.com, где хранится репозиторий проекта, зайти в Marketplace, выбрать категорию Continuous Integration и подкатегорию – Mobile CI.
2) В появившемся меню выбрать App Center (рисунок 2.5).

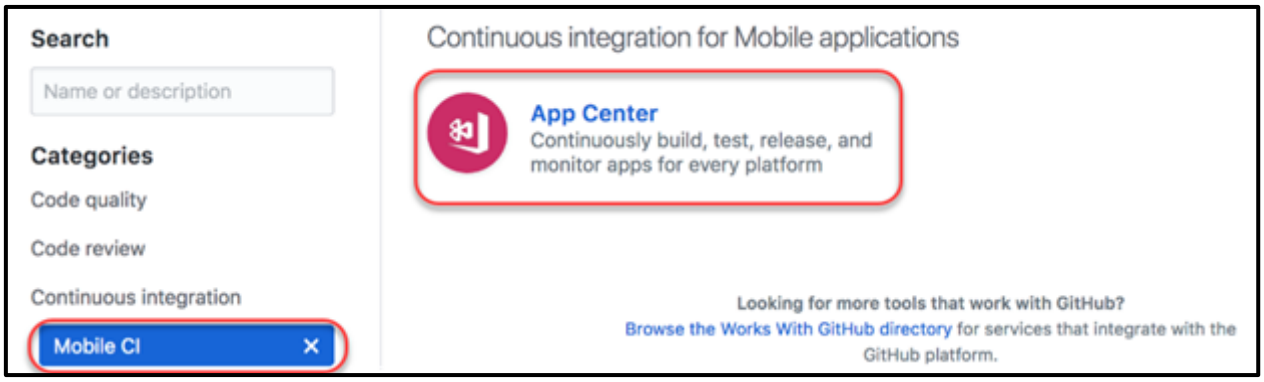

Рисунок 2.5 – 2 шаг внедрения сервиса App Center

3) Выбрать ценовой план и нажать на кнопку «Complete order and begin installation» (рисунок 2.6).

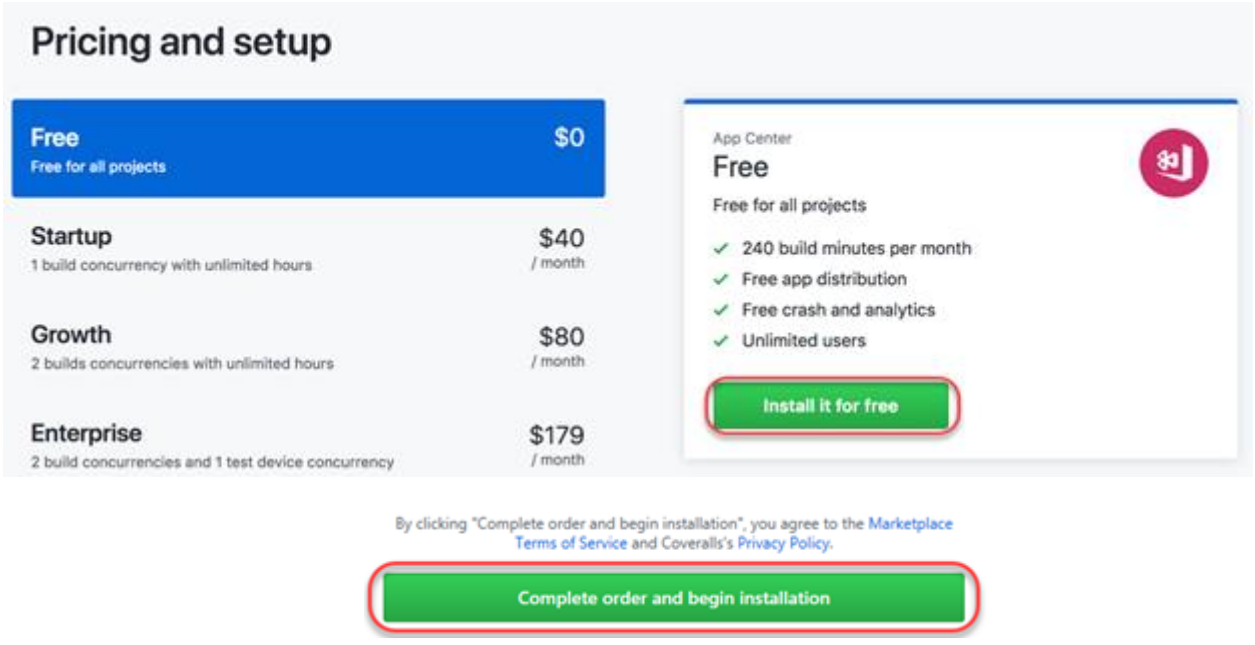

Рисунок 2.6 – 3 шаг внедрения сервиса App Center

4) Выбрать репозиторий проекта «Классная.Москва» и нажать «Install» (рисунок 2.7).
5) На сайте https://appcenter.ms выбрать появившийся репозиторий проекта, нажать кнопку Add (рисунок 2.8).

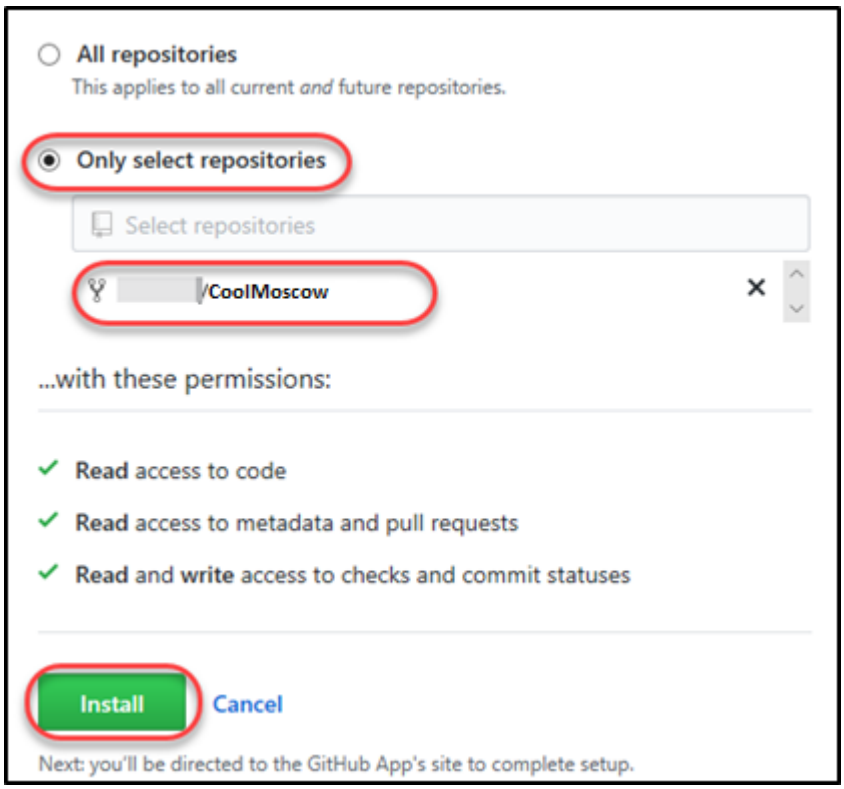

Рисунок 2.7 – 4 шаг внедрения сервиса App Center

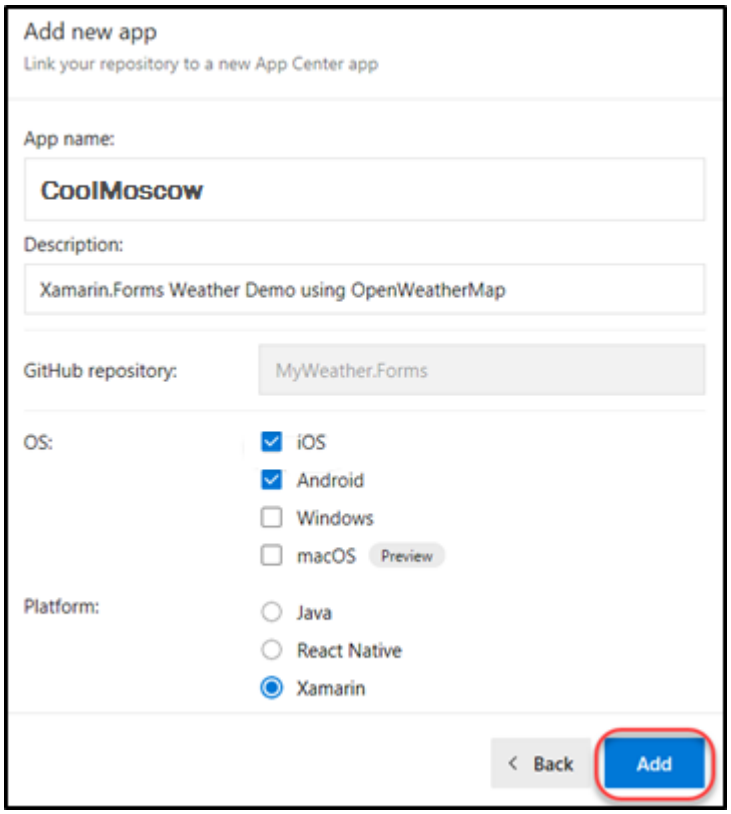

Рисунок 2.8 – 5 шаг внедрения сервиса App Center

6) Настроить наборы устройств, которые будут использоваться при тестировании.

7) В TeamCity добавить в шаг построения автоматический запуск тестов App Center, чтобы при изменении версии приложения, приложение сразу бы тестировалось на реальных устройствах.

Следующим шагом необходимо написать сценарии тестов, которые и будут использоваться при тестировании в App Center.

Фреймворки для UI-тестирования

App Center поддерживает несколько фреймворков, с помощью которых созданы сценарии тестов. Это Calabash (Microsoft), Espresso (Google), XCUITest (Apple), Appium (Open Source), Xamarin.UITest (Microsoft). Поэтому необходимо определить, какой фреймворк использовать для написания тестов.

Calabash – это кроссплатформенный фреймворк с открытым исходным кодом для автоматизации приемочного тестирование Android и iOS приложений. Разработкой и поддержкой занимается компания Microsoft.

Calabash состоит из библиотек, которые дают возможность тесту взаимодействовать с нативными и гибридными приложениями. Взаимодействие заключается в виде реализованных действиях пользователя. Такими действиями могут быть:

- Жесты. Касания и жесты (tap, swipe, rotate и т.п.).
- Проверки. Например, на экране должна присутствовать кнопка «Login» на экране.
- Получения снимка с экрана мобильного устройства

Также в ближайшее время Microsoft перестанет поддерживать данный фреймворк.

Espresso – это фреймворк тестирования пользовательского интерфейса Android-приложений, поддерживаемый Google. Данный фреймворк основан на Java, интегрирован в среду разработки Android Studio и имеет тестовый рекордер, который помогает быстро начать тестирование. Подобно Calabash, он основан на базе инструментов для Android. Поэтому в Espresso нет поддержки межплатформенного тестирования.

XCUITest является официальным фреймворком для тестирования пользовательского интерфейса iOS-устройств. Он поддерживает версии iOS 9 и выше. Тесты могут быть записаны на языках Swift или Objective-C.

Подобно UIAutomation от Apple, XCUITest основана на системе доступности iOS, например, идентификация кнопок через признаки доступности, идентификаторы и метки. XCUITest интегрирован в Xcode и имеет тестовый рекордер.

XCUITest – мощная инфраструктура, но все еще немного страдает от недостатка зрелости. Однако фреймворк активно обновляется и совершенствуется. И поскольку он создан только для iOS-устройств, в нем нет поддержки межплатформенного тестирования.

Appium – это кроссплатформенный фреймворк для тестирования пользовательского интерфейса iOS, Android и Windows устройств. Appium использует архитектуру клиент-сервер, которая позволяет использовать несколько «клиентских» языков и тестовых фреймворков для создания тестов. Для каждого поддерживаемого языка есть клиентская библиотека Appium, которая используется для написания тестов на языке программирования. Его архитектура основана на Selenium: популярном инструменте тестирования работы браузера.

Естественная целевая аудитория для Appium - это тестировщики, которые имеют опыт написания тестов с использованием Selenium.

Xamarin.UITest – это фреймворк для тестирования пользовательского интерфейса iOS и Android устройств. Он основан на той же программной инфраструктуре, что и Calabash. Важно отметить, что, хотя Calabash устарел, Xamarin.UITest продолжает полностью поддерживаться и развиваться командой Xamarin в Microsoft. Так же, как Calabash, Xamarin.UITest работает с любым приложением, независимо от того, какая технология была использована для его создания (т.е. поддерживаются Xcode/Swift/Objective-C, Android/Java, Cordova, Xamarin).

Xamarin.UITest имеет тот же набор функций, что и Calabash, и так же прост в использовании. Существует также поддержка IDE для Xamarin.UITest с помощью шаблонов в Visual Studio и Visual Studio для Mac. Целевой аудиторией данного фреймворка являются разработчики, работающие с платформой Xamarin.

Поскольку в компании «Инфиннити» мобильные приложения разрабатываются с использованием платформы Xamarin одновременно для операционных систем Android и iOS, наиболее оптимальным выбором будет использование фреймворка Xamarin.UITest.

2.9 Разработка дополнительного модуля

На данный момент, для запуска тестов необходимо написать специальную команду в командную строку. Для генерации этой команды необходимо зайти на сайт https://appcenter.ms и выбрать подходящие настройки тестирования. Это не всегда удобно, поскольку зачастую необходимо проводить тесты с одними и теми же настройками, а открывать каждый раз браузер и командную строку не всегда удобно.

Поэтому для более удобного доступа к проведению тестирования была разработана программа на языке Python. Исходный код программы приведен в приложении А. Сценарий работы с ней:

Основной успешный сценарий:

1. Пользователь открывает главное меню программы (рисунок 2.9);
2. Пользователь выбирает подменю «Начать тестирование»
3. Пользователь выбирает настройки для тестов: подходящий набор устройств, системный язык устройств, тестовый фреймворк, на котором написаны сценарии тестов и сами сценарии (рисунок 2.10);
4. Пользователь нажимает кнопку «Начать тестирование»;
5. Программа уведомляет пользователя об успешном начале тестирования (рисунок 2.11);

6. После проведения всех тестов пользователь выбирает подменю «Результаты тестирования», в котором он может увидеть результаты последних тестов: ID теста, дату начала проведения, приложение, которое тестировалось, общее время тестирования, количество устройств, на которых проходили тесты, количество устройств, успешно прошедших данный тест, количество устройств, не прошедших тест и максимальную нагрузку на память устройства (рисунок 2.13).

Расширения (или альтернативные потоки):

4а. Возникла ошибка при запуске тестирования.

1. Программа уведомляет пользователя об отмене запуска тестов и покажет ошибку (рисунок 2.12);
2. Пользователь исправляет проблемы и запускает тесты заново.

5а. Необходимый тест еще не выполнен.

1. Программа показывает, что данный тест находится в стадии выполнения (рисунок 2.14);
2. После проведения всех тестов пользователь сможет увидеть результаты.

Также при первом запуске программы необходимо выполнить ее настройку (рисунок 2.15): заполнить имя пользователя App Center, название приложения и API-токен, который можно получить на сайте https://appcenter.ms. В дальнейшем эти настройки можно изменить, нажав на значок шестеренки.

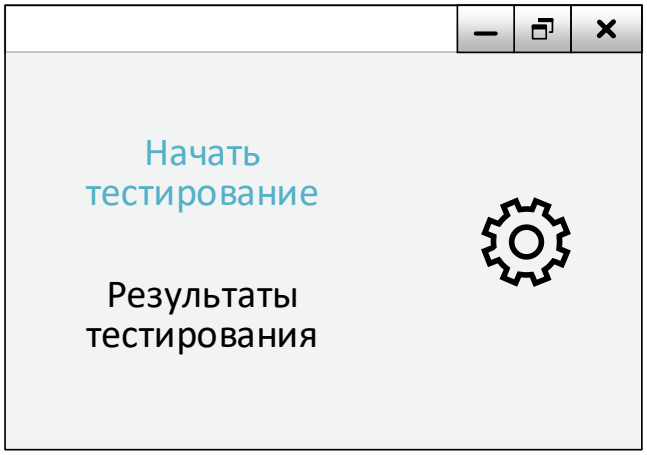

Рисунок 2.9 – Главное меню программы

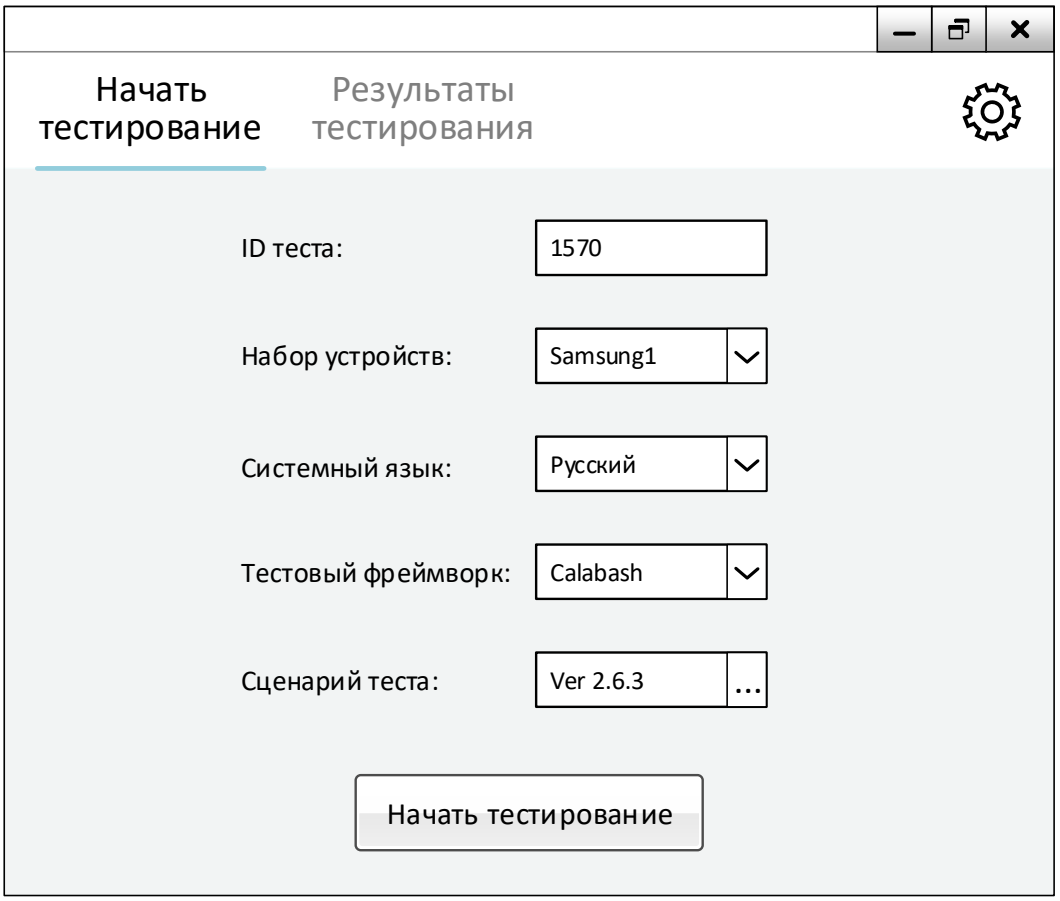

Рисунок 2.10 – Интерфейс меню «Начать тестирование»

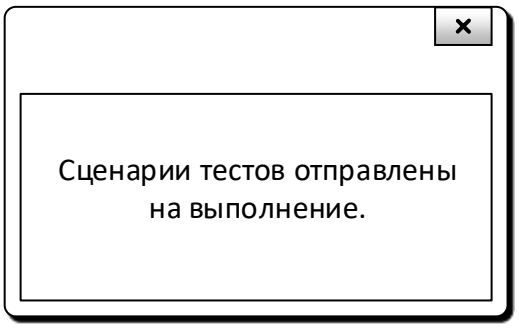

Рисунок 2.11 – Интерфейс оповещения о проведении нового тестирования

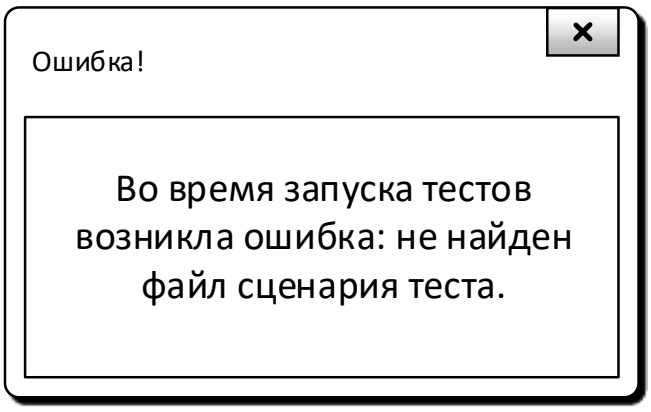

Рисунок 2.12 – Интерфейс оповещения об ошибке запуска тестов

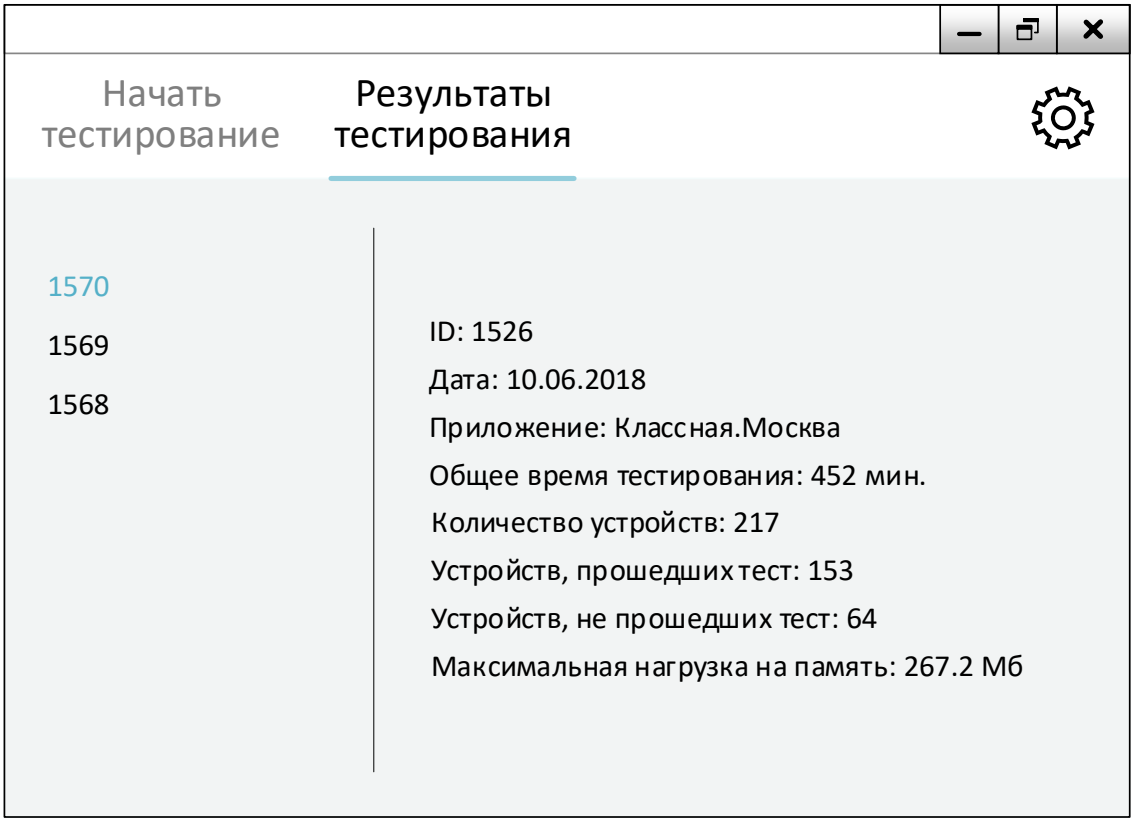

Рисунок 2.13 – Интерфейс меню «Результаты тестирования»

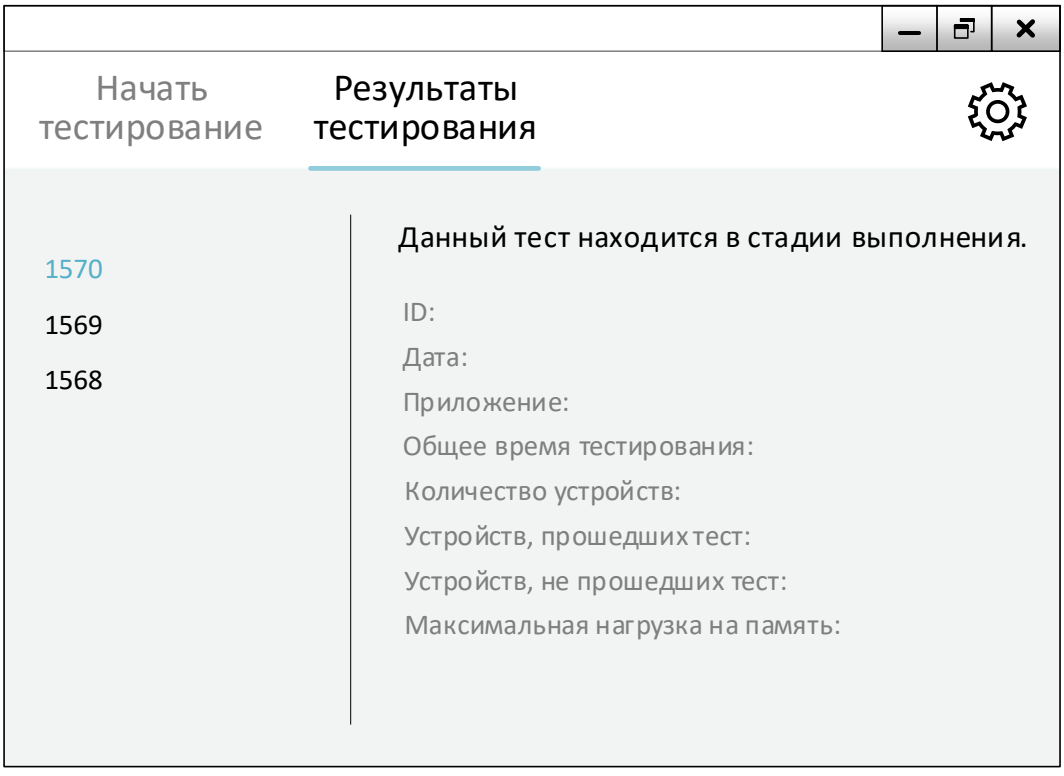

Рисунок 2.14 – Интерфейс меню «Результаты тестирования» во время выполнения теста

Рисунок 2.15 – Интерфейс меню «Настройки»

2.10 Техническая реализация

При реализации проекта алгоритм процесса тестирования практически не изменится:

1. Разработчик пишет программный код приложения в среде MS Visual Studio и передает его в центральный репозиторий под управлением системы управления версиями GIT.
2. Сервер непрерывной интеграции TeamCity опрашивает центральное хранилище на наличие обновлений.
3. Система управления версиями сообщает серверу о наличии обновлений.
4. Сервер TeamCity формирует очередь тестируемого программного кода и дает команду своим агентам произвести тестирование, а также отправляет запрос на сервер App Center для проведения тестов на реальных устройствах.
5. Агенты TeamCity загружают из центрального репозитория обновления, а затем с помощью программных средств осуществляют модульное и UI тестирование приложения.

6. После завершения тестирования агенты сообщают серверу TeamCity о результатах выполнения тестов.
7. App Center также сообщает серверу TeamCity о результатах выполнения тестов после их завершения.
8. Сервер TeamCity сообщает разработчику в интерфейс среды Visual Studio какие тесты прошли успешно, а какие нет.
9. Разработчик фиксирует в центральном репозитории успешно прошедший тестирование программный код.

Для реализации такого процесса необходимы следующие компоненты:

1. Компьютер разработчика

   Системные требования:

   - Операционная система: Windows 7/8/10, Windows Server 2012, 2016
   - Двухъядерный процессор с тактовой частотой от 1,8 GHz
   - Оперативная память от 2 ГБ
   - 130 ГБ свободного места на жестком диске
   - Видеоадаптер с минимальным разрешением 720p (1280 на 720 пикселей)
   - Сетевая карта с пропускной способностью от 100 Мбит/с
   - Наличие интегрированной среды разработки Visual Studio 2017
   - Наличие фреймворка Xamarin
2. Сервер TeamCity

   Системные требования:

   - Операционная система: Linux, macOS, Windows 7/8/10, Windows Server 2008, 2012, 2016
   - Двухъядерный процессор с тактовой частотой от 3.2 GHz
   - Оперативная память от 4 ГБ
   - Жесткий диск 1 ТБ
   - Сетевая карта с пропускной способностью от 100 Мбит/с

- Наличие TeamCity

3. Компьютер для агентов TeamCity

   Системные требования:

   - Операционная система: Windows 7/8/10, Windows Server 2012, 2016
   - Процессор с тактовой частотой от 1.8 GHz
   - Оперативная память от 2 ГБ
   - Жесткий диск 1 ТБ
   - Сетевая карта с пропускной способностью от 100 Мбит/с
   - Наличие TeamCity
   - Наличие интегрированной среды разработки Visual Studio 2017

Для визуализации аппаратной архитектуры используется диаграмма развертывания (рисунок 2.16). Диаграмма развертывания – диаграмма, на которой представлены узлы выполнения программных компонентов реального времени, а также процессов и объектов.

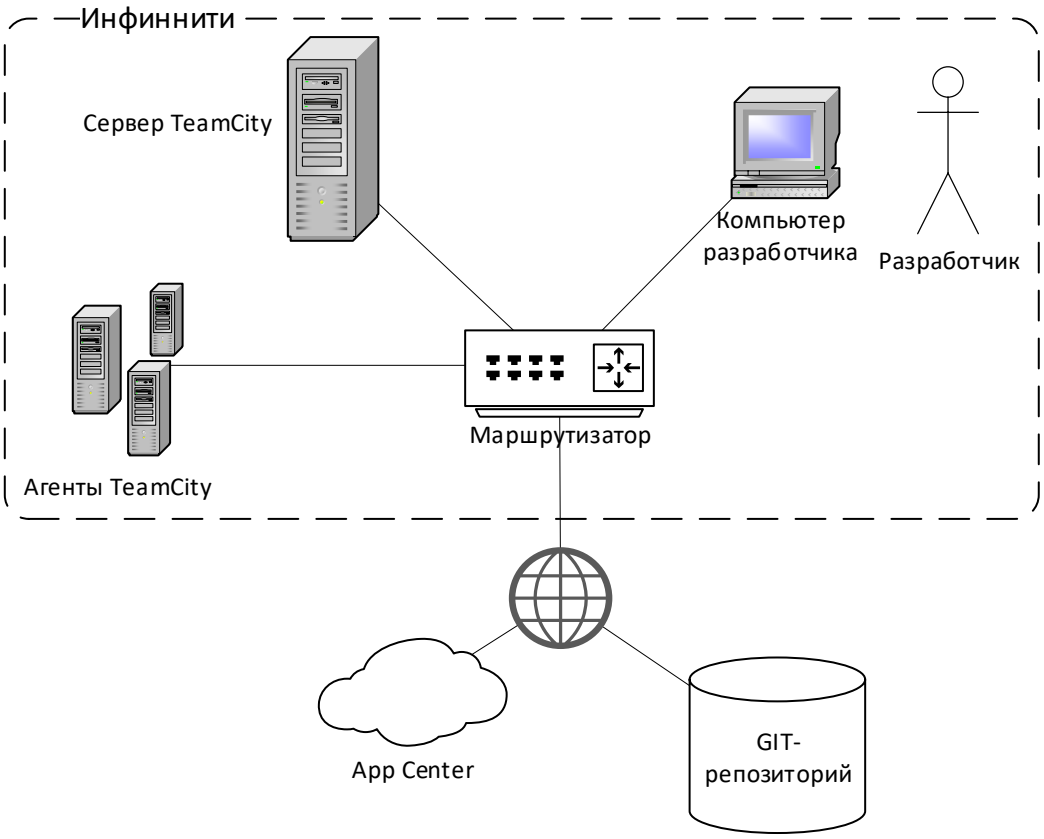

Рисунок 2.16 – Диаграмма развертывания

В «Инфиннити» уже практически полностью реализован данный процесс, а новые задачи выполняются на облачном сервере App Center. Поэтому программно-аппаратные средства в компании соответствуют всем необходимым требованиям для реализации архитектуры приложения.

Выводы по главе 2

Во второй главе был описан сервис Классная Москва, бизнес-процесс тестирования мобильных приложений. Также были рассмотрены способы тестирования мобильных приложений. Был предложен проект совершенствования тестирования с помощью внедрения сервиса App Center. Также была разработана программа, упрощающая работу с данным сервисом.

ГЛАВА 3. ОЦЕНКА ЭФФЕКТИВНОСТИ ПРОЕКТА

3.1 Календарный план проекта

Для выявления и осознания целей, состава и содержания проекта, организации планирования и контроля процессов осуществления проектов необходимо определить и построить структуру работ проекта.

Этап 1. Планирование

- Определение проблем в тестировании – необходимо определить и проанализировать имеющиеся проблемы в проведении тестировании мобильных приложений, определить причины их появления.
- Выбор ПО для реализации проекта – после определения проблем в тестировании, необходимо выбрать программный продукт (или связку продуктов) для решения проблемы, проанализировать имеющийся на предприятии программно-аппаратный комплекс, насколько он соответствует требованиям проекта.
- Определение финансовых затрат – после выбора программных средств, необходимо определить финансовые затраты на проект.
- Составление плана работ – после определения финансовых затрат, необходимо составить план выполнения работ, где следует учесть список задач, очередность их выполнения, а также определить состав команды проекта.
- Утверждение плана работ – после составления плана работ, необходимо его согласовать с руководителем департамента разработки.

Этап 2. Работа с ПО

- Установка и настройка программных средств – необходимо установить и сконфигурировать программное обеспечение для реализации целей проекта.
- Тестирование ИС – после установки и настройки программных средств, необходимо протестировать работу ИС целиком.

- Доработка ИС – параллельно с тестированием системы, необходимо дорабатывать ее с целью валидности поставленным требованиям, достижения удобства использования.

Этап 3. Обучение сотрудников.

- Обучение сотрудников отделов – этап предполагает, что сотрудники отделов должны быть посвящены в суть изменений в их работе после начала использования автоматизированных тестовых методик, ознакомлены с основными принципами реализуемого подхода.

Этап 4. Ввод системы в эксплуатацию.

- Опытная эксплуатация системы – самая долгая по длительности работа, предполагает использование автоматического функционального тестирования с целью определения степени удобства, надежности и эффективности подхода.
- Составление регламента взаимодействия – после апробации системы, необходимо регламентировать процесс автоматического тестирования.
- Приемка системы – после составления регламента взаимодействия, его надо согласовать с руководителем департамента разработки и утвердить.

На рисунке 3.1 приведен перечень работ проекта. Расписание проекта (диаграмма Ганта) приведено на рисунке 3.2.

На рисунке 3.3 перечислены все ресурсы, необходимые для реализации проекта.

Статистика проекта приведена на рисунке 3.4. Длительность проекта составит 35 дней, трудозатраты – 472 часа, финансовые затраты – 124000 р.

		Режим задачи	Название задачи	Длительнс	Начало	Окончани	Пред	Названия ресурсов
1			**Планирование**	**8 дней**	**Пн 02.07.18**	**Ср 11.07.18**		
2			Определение проблем в тестировании	1 день	Пн 02.07.18	Пн 02.07.18		Системный аналитик[200%];Тестировщик
3			Выбор ПО для реализации проекта	2 дней	Вт 03.07.18	Ср 04.07.18	2	Разработчик
4			Определение финансовых затрат	2 дней	Чт 05.07.18	Пт 06.07.18	3	Финансист
5			Составление плана работ	2 дней	Пн 09.07.18	Вт 10.07.18	4	Системный аналитик
6			Утверждение плана работ	1 день	Ср 11.07.18	Ср 11.07.18	5	Руководитель департамента разработки
7			**Работа с ПО**	**8 дней**	**Чт 12.07.18**	**Пн 23.07.18**		
8			Установка и настройка программных средств	1 день	Чт 12.07.18	Чт 12.07.18	6	Разработчик[200%]
9			Тестирование ИС	5 дней	Пт 13.07.18	Чт 19.07.18	8	Тестировщик[200%]
10			Доработка ИС	2 дней	Пт 20.07.18	Пн 23.07.18	9	Разработчик[150%]
11			**Обучение сотрудников**	**2 дней**	**Вт 24.07.18**	**Ср 25.07.18**		
12			Обучение сотрудников	2 дней	Вт 24.07.18	Ср 25.07.18	10	Разработчик
13			**Ввод системы в эксплуатацию**	**17 дней**	**Чт 26.07.18**	**Пт 17.08.18**		
14			Опытная эксплуатация системы	12 дней	Чт 26.07.18	Пт 10.08.18	12	Разработчик;Тестировщик
15			Составление регламента взаимодействия	3 дней	Пн 13.08.18	Ср 15.08.18	14	Разработчик;Тестировщик
16			Приемка системы	1 день	Чт 16.08.18	Чт 16.08.18	15	Руководитель департамента разработки
17			Ввод в эксплуатацию	1 день	Пт 17.08.18	Пт 17.08.18	16	Разработчик

Рисунок 3.1 – Перечень работ проекта

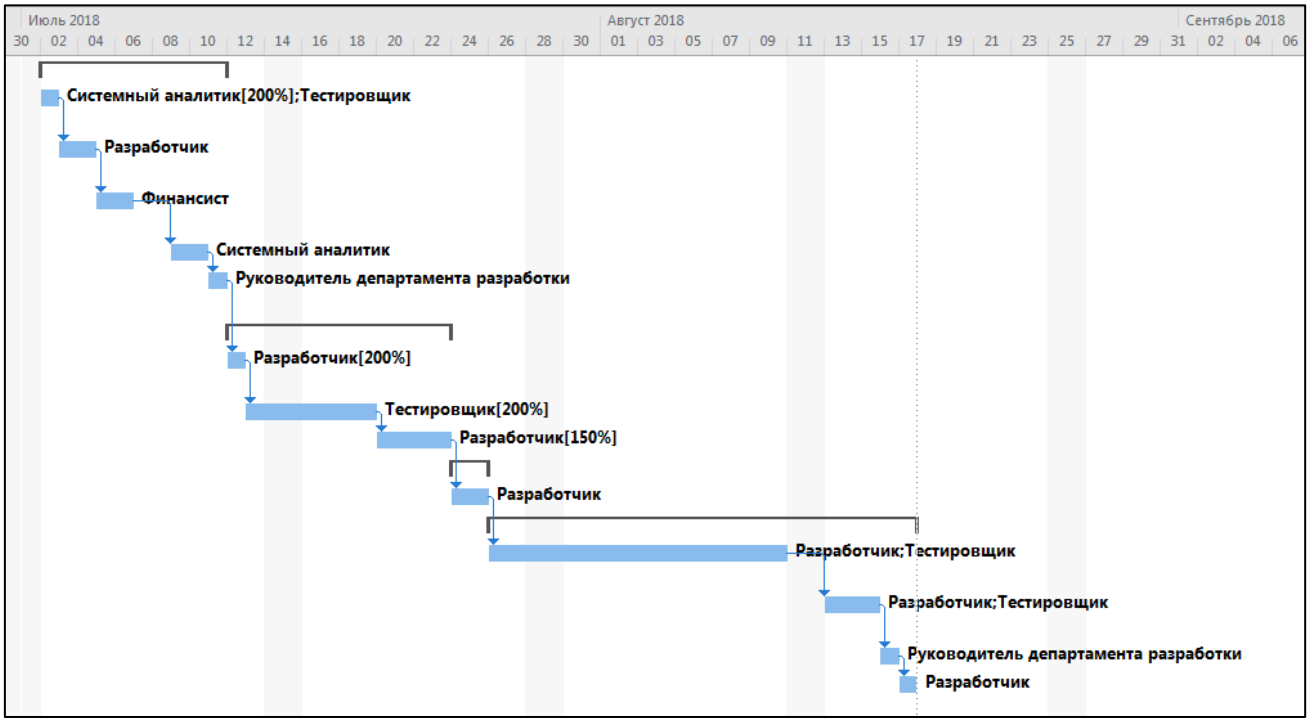

Рисунок 3.2 – Диаграмма Ганта

		Название ресурса	Тип	Единицы измерения материалс	Краткое названи	Группа	Макс. единиц	Стандартна ставка	Ставка сверхурочн	Затраты на использ.	Начисление	Базовый календарь
1		Системный аналитик	Трудовой		С		200%	300,00 ₽/ч	450,00 ₽/ч	0,00 ₽	Пропорциональ	Стандартный
2		Руководитель департамента разработки	Трудовой		Р		100%	500,00 ₽/ч	750,00 ₽/ч	0,00 ₽	Пропорциональ	Стандартный
3		Разработчик	Трудовой		Р		300%	300,00 ₽/ч	450,00 ₽/ч	0,00 ₽	Пропорциональ	Стандартный
4		Тестировщик	Трудовой		Т		200%	200,00 ₽/ч	300,00 ₽/ч	0,00 ₽	Пропорциональ	Стандартный
5		Финансист	Трудовой		Ф		100%	300,00 ₽/ч	450,00 ₽/ч	0,00 ₽	Пропорциональ	Стандартный

Рисунок 3.3 – Лист ресурсов проекта

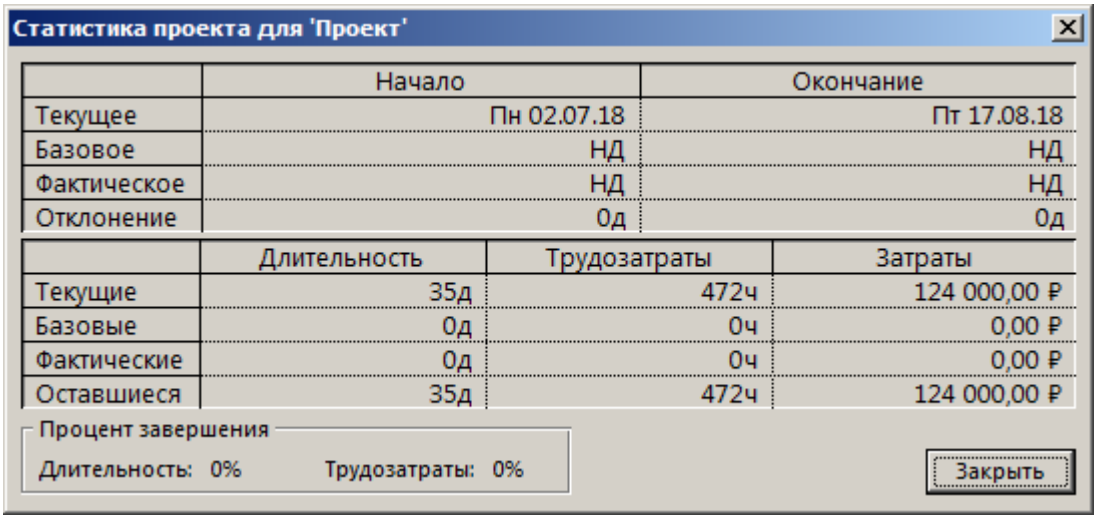

Рисунок 3.4 – Статистика проекта

3.2 Управление рисками

3.2.1 Идентификация рисков

Работа с рисками проекта начинается с их идентификации. В качестве основных выявленных рисков настоящего проекта выделены следующие:

1. Переход к другой среде разработки.

   При отказе от использования MS Visual Studio, результаты проекта нельзя будет использовать в другой среде разработки. В случае реализации данного риска, предприятие понесет расходы на поиск и внедрение новых программных средств.

2. Риск прекращения поддержки разработчиком используемых в проекте программных средств.

   Если разработчики прекратят поддержку продукта и выпуск новых версий, это повлечет за собой вероятность невозможности интеграции программных средств между собой.

3. Неточное планирование времени и ресурсов.

   При неточном распределении времени и ресурсов для реализации проекта могут появиться следующие последствия: отставание по срокам, потеря прибыли, непредвиденные траты.

4. Разногласия внутри проектной команды.

Разногласия внутри команды проекта может привести к срыву сроков проекта и возникновению ошибок в ходе его реализации.

5. Недостаточная квалификация участников проектной команды.

   В случае реализации риска могут быть нарушены сроки проекта, могут возникнуть ошибки, устранение которых потребует привлечение дополнительных ресурсов.

6. Загруженность сотрудников.

   Увеличение нагрузки на сотрудников во время внедрения системы, совмещение обучения и работы могут привести к простоям в производстве и превышению сроков проекта.

7. Переоценка возможностей ИС

   Неточный анализ возможностей ИС и анализ функциональной полноты может привести к переоценке возможностей системы и как к непредвиденным тратам, так и к срыву всего проекта.

3.2.2 Качественные анализ рисков

После идентификации рисков необходимо провести качественный анализ.

Для начала необходимо определить вероятность возникновения рисков, для этого рассмотрим шкалу вероятности в таблице 3.1.

Таблица 3.1 – Оценка вероятности

Оценка вероятности возникновения риска				
Интервал вероятностей	**Значение вероятности, используемое для вычислений**	**Словесная формулировка**	**Числовая оценка**	**Номер риска**
От 1% до 18%	9%	маловероятно	1	1, 2, 4
От 19% до 38%	28%	скорее нет	2	3, 7
От 39% до 58%	48%	50-50	3	5
От 59% до 78%	68%	весьма правдоподобно	4	-
От 79% до 99%	88%	почти наверняка	5	6

Затем нужно оценить последствия рисков по финансовой шкале в таблице 3.2.

Таблица 3.2 – Финансовая шкала

Шкала для оценки последствий риска, измеряемого в деньгах		
Оценка	**Денежное выражение**	**Номер риска**
1	До 5,000р	-
2	5,000р – 30,000р	4, 5
3	31,000р – 60,000р	2, 3, 6
4	61,000р – 120,000р	1
5	Более 120,000р	7

По полученным данным из шкал составим качественный анализ рисков проекта (рисунок 3.5).

Рисунок 3.5 – Качественный анализ рисков

Таким образом, идентифицировав риски и проведя их качественный анализ, мы выявили наиболее существенные риски проекта, на которые риски необходимо обратить особое внимание.

Наиболее опасным рискам в результате анализа являются Р6 – «Загруженность сотрудников».

3.2.3 Количественный анализ рисков

Исходя из качественного анализа рисков был выявлен риск, оказывающий наибольшее влияние. Теперь необходимо провести его количественный анализ.

Риск «Загруженность сотрудников». Вероятность – 0,28, возможные финансовые последствия – 120000 руб. Причины – необучаемость сотрудника, нехватка кадров. Рассмотрим дерево решений для данного риска на рисунке 3.6.

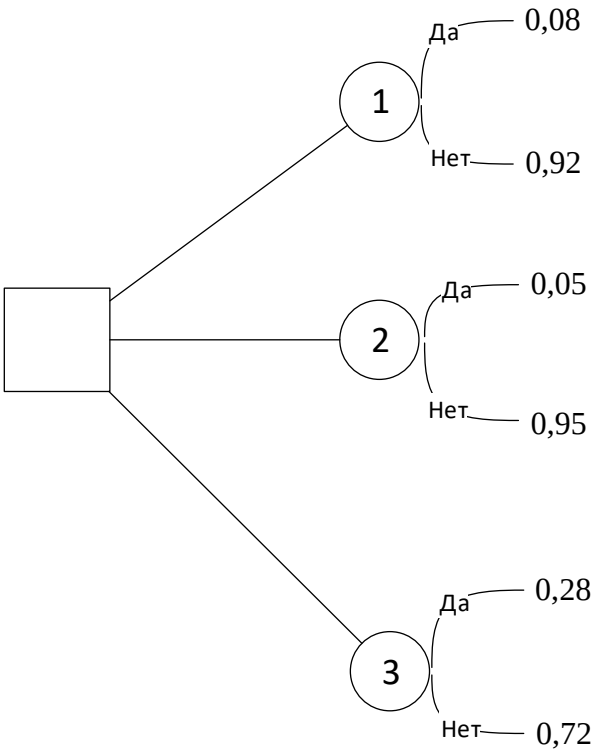

Рисунок 3.6 – Дерево решений риска «Загруженность сотрудников»

Для минимизации данного риска можно предложить следующие мероприятия:

1) Наём дополнительных сотрудников на временную ставку, что позволит разгрузить сотрудников, занятых обучением. При этом будет дополнительно затрачено 20 тыс. руб. Вероятность реализации риска в таком случае составит 8% с соответствующими затратами 140 тыс. руб. (120+20), а вероятность невыполнения риска – 92% с затратами 20 тыс. руб.;

2) Полностью обновить штат сотрудников. При этом будет дополнительно затрачено 40 тыс. руб. Вероятность реализации риска в таком случае составит 8% с соответствующими затратами 160 тыс. руб. (120+40), а вероятность невыполнения риска – 92% с затратами 40 тыс. руб.;
3) Не осуществлять никаких мероприятий. Тогда вероятность реализации риска составит 28% с соответствующими затратами 120 тыс. руб., а вероятность невыполнения риска – 72% с нулевыми затратами;

Теперь рассчитаем финансовую эффективность решений.

Управленческое решение 1 – «Наём дополнительных сотрудников на временную ставку»: 0,08*140000+0,92*20000 = 29600 р.

Управленческое решение 2 – «Обновление штата сотрудников»: 0,05*160000+0,95*40000 = 46000 р.

Управленческое решение 3 – «Не осуществлять никаких мероприятий»: 0,28*120000+0,72*0 = 33600 р.

Таким образом, управленческое решение «Наём дополнительных сотрудников на временную ставку» по нейтрализации риска «Загруженность сотрудников» является наиболее оптимальным по цене.

3.3 Финансовый анализ эффективности

3.3.1 Совокупная стоимость владения

Для того чтобы определить экономическую эффективность проекта, необходимо произвести соотношение финансовых затрат на реализацию проекта результатов проекта, обеспечивающих требуемую норму доходности.

Рассмотрим основные статьи затрат, которые понесет предприятие в ходе реализации проекта.

Тестирование с помощью сервиса App Center осуществляется по подписочной основе. Стоимость подписки - 6000 р/месяц. Весь остальной имеющийся аппаратный комплекс полностью отвечает требованиям проекта, а программное обеспечение либо уже установлено, либо бесплатно.

Затраты на персонал: реализация проекта будет осуществлена полностью трудовыми ресурсами предприятия. Эти затраты составят 124000 р.

Последующего сопровождения система не требует, поэтому расходы на обслуживание проект не предполагает.

3.3.2 Определение ставки дисконтирования

Ставка дисконтирования – это процентная ставка, используемая для пересчёта будущих потоков доходов в единую величину текущей стоимости. По методу кумулятивного построения величина ставки дисконтирования определяется как сумма безрисковой ставки и надбавок на риск: $i=G+\sum R(i)$, где G – безрисковая ставка (7,25%), $\sum R(i)$ – сумма возможных рисков. [7]

Возможные виды рисков:

1. Неправильно определены сроки работы 3%
2. Отказ оборудования или его поломка 2%
3. Высокая занятость заказчика 5%

Итоговая ставка дисконтирования равна: $i = 7,25 + 3 + 2 + 5 = 17,25\%$

3.3.3 Модель денежных потоков

Производится разработка и внедрение модуля тестирования на реальных устройствах. Стоимость внедрения – 124000 рублей, стоимость сервиса тестирования – 6000 рублей в месяц, срок внедрения – полтора месяца, производится с помощью аутсорсинга.

Внедрение модуля приведет к сокращению одной ставки тестировщика с заработной платой в 30000 рублей в месяц.

Ставка дисконтирования $i = 17,25\%$. Срок проекта 9 месяцев.

Расчет налогооблагаемой базы и изменение налога на прибыль приведены в таблице 3.3.

Таблица 3.3 – Расчет налогов

Изменение налогооблагаемой базы	Изменение налога на прибыль
-124000/36 - 6000 + 30000*1,302 = 29615,6	29615,6*0,2 = 5923,1

Поскольку проект рассчитывается по месяцам, ставка дисконтирования была пересчитана из годовой в месячную в соответствии с формулой:

$i_{\text{мес}}$=((1+0,1725)^(1/12))-1=0,013

Чистая текущая стоимость проекта была рассчитана в модели денежных потоков (таблица 3.4).

Таблица 3.4 – Модель денежных потоков

Период	Доходы	Расходы	ЧДП	ДМ	ЧДД	ЧТС
1	0	82666,7	-82666,7	0,987	-81592	-81592
2	0	14196,4	-14196,4	0,974	-13827,3	-95419,3
3	27136,9	0	27136,9	0,962	26105,7	-69313,6
4	27136,9	0	27136,9	0,95	25780,1	-43533,5
5	27136,9	0	27136,9	0,937	25427,3	-18106,2
6	27136,9	0	27136,9	0,925	25101,6	7321,1
7	27136,9	0	27136,9	0,913	24776	32097,1
8	27136,9	0	27136,9	0,901	24450,3	56547,4
9	27136,9	0	27136,9	0,89	24151,8	80699,2

На рисунке 3.7 представлен график денежных потоков.

NPV (чистая приведенная стоимость) = последний ЧТС = ∑ЧДД

Из таблицы можно сделать следующие выводы:

NPV (Чистая приведенная стоимость) = 80699,2 рублей > 0, значит проект выгоден.

Срок окупаемости проекта ($Т_{ок}$) = $5+\frac{|-18106,2|}{25101,6}$ = 5,72.

Рисунок 3.7 – График денежных потоков

Дисконтируемый индекс прибыльности $(PI_{\mathrm{d}}) = \frac{26105,7+25780,1+25427,3+25101,6+24776+24450,3+24151,8}{81592+13827,3} = 1,84$. Данная цифра означает, что на каждую затраченную денежную единицу получается 0,84 чистой прибыли.

Индекс прибыльности проекта больше 1, значит он рентабельный. Вложения в проект окупятся в течение срока проекта.

Выводы по главе 3

В результате третьей главы был разработан календарный план внедрения проекта, идентифицированы и оценены риски при внедрении, оценена экономическая эффективность проекта с их учетом. Показатели эффективности доказали, что инвестиции рентабельны и проект является прибыльным.

ЗАКЛЮЧЕНИЕ

В данном проекте была проанализирована деятельность компании ООО «Инфиннити». Были определены внешние и внутренние факторы, которые оказывают влияние на деятельность компании и препятствуют достижению основных целей. Были выявлены основные проблемы компании и предложен путь их решения с помощью информационных технологий.

В результате проведенного исследования проблем была построена диаграмма бизнес-процесса тестирования мобильных приложений. Также были рассмотрены способы тестирования мобильных приложений. Был предложен проект совершенствования тестирования с помощью внедрения сервиса App Center. Также была разработана программа, упрощающая работу с данным сервисом. Проработан пользовательский интерфейс.

Оценка эффективности проекта показала, что проект принесет выгоду для компании ООО «Инфиннити». Был предложен календарный план по осуществлению разработки и внедрению сервиса тестирования мобильных приложений, рассмотрены и учтены риски проекта. Показатели эффективности доказали, что инвестиции рентабельны и приемлемы, проект принесет прибыль компании.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Инструкция по подключению сервиса App Center к серверу непрерывной интеграции– [Электронный ресурс] – Код доступа: https://docs.microsoft.com/ru-ru/xamarin/tools/ci/
2. Официальный сайт компании ООО «Инфиннити» – [Электронный ресурс] – Код доступа: http://www.infinnity.ru/
3. Официальный сайт компании Xamarin – [Электронный ресурс] – Код доступа: https://www.xamarin.com/
4. Официальный сайт сервиса «Классная.Москва» – [Электронный ресурс] – Код доступа: https://какая.классная.москва
5. Официальный сайт сервиса App Center – [Электронный ресурс] – Код доступа: https://appcenter.ms/
6. Сэм Канер, Джек Фолк; Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений, 2001. – 544 с.
7. Шепталин, Г.А. Информационные технологии в управлении проектами: учебное пособие/ Г.А. Шепталин, Н.Э. Решетова, А.Г. Шепталин. – Челябинск: Издательский центр ЮурГУ, 2009. – 121 с.
8. Шепталин, Г.А. Информационный менеджмент: учебное пособие. – Челябинск/ Г.А. Шепталин. Издательский центр ЮУрГУ, 2012 г. – 143 с.
9. Шепталина, Л.И. Исследование систем управления: конспект лекций//Л.И. Шепталина. − Челябинск: Издательский центр ЮУрГУ, 2010 г. − 148 с.

ПРИЛОЖЕНИЕ А

Исходный код разработанной программы

```
from tkinter import *
from tkinter import messagebox as mb
import requests
import json

owner_name = ''
app_name = ''
api_token = ''

root = Tk()

def clear():
    list = root.grid_slaves()
    for l in list:
        l.destroy()

def start():
    button_start = Button(text="Начать \nтестирование", height=4, font=("Calibri", 24))
    button_report = Button(text="Результаты \nтестирования", height=4, font=("Calibri", 24))
    c = Canvas(root, width=250)
    c.create_oval(70, 70, 180, 180, width=2)
    button_start.grid(row=0, column=0)
    button_report.grid(row=1, column=0)
    c.grid(row=0, column=1, rowspan=2)
    button_start.bind('<Button-1>', start_test)

def start_test(event):
    clear()
    button_start = Button(text="Начать \nтестирование", height=2, font=("Calibri", 20))
    button_report = Button(text="Результаты \nтестирования", height=2, font=("Calibri", 20))
    c = Canvas(root, width=250, height=40)
    c.create_oval(205, 5, 235, 35, width=2)
    button_start.grid(row=0, column=0)
    button_report.grid(row=0, column=1)
    c.grid(row=0, column=2, columnspan=2)
    Label(text="ID теста:", height=2, font=("Calibri", 14)).grid(row=1, column=0, sticky=E,
padx=20, columnspan=2)
    Label(text="Набор устройств:", height=2, font=("Calibri", 14)).grid(row=2, column=0,
sticky=E, padx=20, columnspan=2)
    Label(text="Системный язык:", height=2, font=("Calibri", 14)).grid(row=3, column=0,
sticky=E, padx=20, columnspan=2)
    Label(text="Тестовый фреймворк:", height=2, font=("Calibri", 14)).grid(row=4, column=0,
sticky=E, padx=20, columnspan=2)
    Label(text="Сценарий теста:", height=2, font=("Calibri", 14)).grid(row=5, column=0,
sticky=E, padx=20, columnspan=2)
    Entry(root, width=35).grid(row=1, column=2, sticky=W, padx=10, columnspan=2)
    Entry(root, width=35).grid(row=2, column=2, sticky=W, padx=10, columnspan=2)
```

Рисунок А.1 – исходный код программы

```
    Entry(root, width=35).grid(row=3, column=2, sticky=W, padx=10, columnspan=2)
    Entry(root, width=35).grid(row=4, column=2, sticky=W, padx=10, columnspan=2)
    Entry(root, width=35).grid(row=5, column=2, sticky=W, padx=10, columnspan=2)
    button_init_test = Button(text="Начать тестирование", height=2, font=("Calibri", 16))
    button_init_test.grid(row=6, column=0, columnspan=4)
    button_report.bind('<Button-1>', report_test)
    button_init_test.bind('<Button-1>', check)

def report_test(event):
    clear()
    r = test_report(id)
    button_start = Button(text="Начать \nтестирование", height=2, font=("Calibri", 20))
    button_report = Button(text="Результаты \nтестирования", height=2, font=("Calibri", 20))
    c = Canvas(root, width=250, height=40)
    c.create_oval(205, 5, 235, 35, width=2)
    button_start.grid(row=0, column=0)
    button_report.grid(row=0, column=1)
    c.grid(row=0, column=2, columnspan=2)
    Button(text="1570", height=2, font=("Calibri", 14)).grid(row=1, column=0, sticky=W,
padx=20, pady=10)
    Button(text="1569", height=2, font=("Calibri", 14)).grid(row=2, column=0, sticky=W,
padx=20, pady=10)
    Button(text="1568", height=2, font=("Calibri", 14)).grid(row=3, column=0, sticky=W,
padx=20, pady=10)
    Button(text="1567", height=2, font=("Calibri", 14)).grid(row=4, column=0, sticky=W,
padx=20, pady=10)
    f_report = Frame(root)
    Label(f_report, text=f"ID: {r[app_upload_id]}", height=2, font=("Calibri", 14)).grid(row=0,
column=0, sticky=W, padx=10)
    Label(f_report, text=f"Дата: {r[date]}", height=2, font=("Calibri", 14)).grid(row=1,
column=0, sticky=W, padx=10)
    Label(f_report, text=f"Приложение: {r[application]}", height=2, font=("Calibri",
14)).grid(row=2, column=0, sticky=W, padx=10)
    Label(f_report, text=f"Общее время тестирования: {r[totalDeviceMinutes]}", height=2,
font=("Calibri", 14)).grid(row=3, column=0, sticky=W, padx=10)
    Label(f_report, text=f"Количество устройств: {r[total]}", height=2, font=("Calibri",
14)).grid(row=4, column=0, sticky=W, padx=10)
    Label(f_report, text=f"Устройств, прошедших тест: {r[passed]}", height=2, font=("Calibri",
14)).grid(row=5, column=0, sticky=W, padx=10)
    Label(f_report, text=f"Устройств, прошедших тест: {r[failed]}", height=2, font=("Calibri",
14)).grid(row=6, column=0, sticky=W, padx=10)
    Label(f_report, text=f"Макимальная нагрузка на память: {r[peakMemory]}", height=2,
font=("Calibri", 14)).grid(row=7, column=0, sticky=W, padx=10)
    f_report.grid(row=1, column=1, columnspan=3, rowspan=4)
    button_start.bind('<Button-1>', start_test)

def settings(event):
    clear()
    Label(text="Пользователь:", height=2, font=("Calibri", 14)).grid(row=0, column=0)
    Label(text="Приложение:", height=2, font=("Calibri", 14)).grid(row=1, column=0)
```

Рисунок А.1 – исходный код программы (продолжение)

```
   Label(text="API-токен:", height=2, font=("Calibri", 14)).grid(row=2, column=0, sticky=W,
padx=10)

   e1 = Entry(root, width=35).grid(row=0, column=1, sticky=W)
   e2 = Entry(root, width=35).grid(row=1, column=1, sticky=W)
   e3 = Entry(root, width=35).grid(row=2, column=1, sticky=W)
   f_report = Frame(root)
   button_ok = Button(f_report, text="OK", height=2, font=("Calibri", 20))
   button_apply = Button(f_report, text="Применить", height=2, font=("Calibri", 20))
   button_cancel = Button(f_report, text="Отмена", height=2, font=("Calibri", 20))
   button_ok.grid(row=0, column=0)
   button_apply.grid(row=0, column=1)
   button_cancel.grid(row=0, column=2)
   f_report.grid(row=3, column=0, columnspan=2)
   button_ok.bind('<Button-1>', apply_settings(e1.get(), e2.get(), e3.get()))

def start_test(id, device_selection, language, test_framework, locale, scenario):
        startOptions = {"test_framework": test_framework, "device_selection": device_selection,
"language": language,"test_parameters": {}}
        headers = {'test_run_id': id, 'startOptions': startOptions, 'owner_name': owner_name,
'app_name': app_name, 'api-token': api-token}
        r =
requests.get(f"https://api.appcenter.ms/v0.1/{app_name}/{owner_name}/app/test_runs/{id}/start
", headers=headers)
        j = json.loads(r.text)

   if r.code == 200:
                mb.showinfo("Запуск тестирования", "Сценарии тестов отправлены на
выполнение.")
        else:
                mb.showerror(f"Ошибка", "Во время запуска тестов возникла ошибка:
{r.info}")

def apply_settings(owner, name, token):
        owner_name = owner
        app_name = name
        api-token = token

def test_report(id):
        r =
requests.get(f"https://api.appcenter.ms/v0.1/{app_name}/{owner_name}/app/test_runs/{id}/state
")
   return json.loads(r.text)

start()
root.mainloop()
```

Рисунок А.1 – исходный код программы (продолжение)