Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

Дисциплина «Основы управления проектами»

Отчет к лабораторной работе № 7

«Разработка плана коммуникаций

проекта и плана управления конфигурациями»

Проект: мобильное приложение для уведомления о приеме лекарств и отслеживания метрик здоровья

Выполнил:

Василения Иван Валерьевич

Принял:

Олькина Елена Викторовна

Орёл, 2025г

1. План сбора информации

Источники информации и методы получения:

* Целевая аудитория (пенсионеры и их семьи):
  + Методы:
    - Анкетирование и фокус-группы для выявления потребностей.
    - Тестовые сессии с прототипом приложения для оценки удобства интерфейса.
  + Источники: Социальные центры, медицинские учреждения, онлайн-сообщества.
* Медицинские эксперты:
  + Методы:
    - Консультации для проверки корректности медицинских рекомендаций и метрик (уровень сахара, давление).
    - Верификация данных о лекарствах.
  + Источники: Врачи-терапевты, геронтологи, фармацевты.
* Технические специалисты:
  + Методы:
    - Сбор требований к совместимости с iOS/Android.
    - Анализ энергопотребления и офлайн-функционала.
  + Источники: Разработчики, тестировщики, архитекторы приложения.
* Рыночные данные:
  + Методы:
    - Анализ конкурентов (функционал, UX, цены).
    - Изучение трендов в мобильных приложениях для здоровья.
  + Источники: App Store, Google Play, отчеты аналитических агентств.
* Юридические аспекты:
  + Методы:
    - Консультации с юристами по GDPR и защите персональных данных.
    - Анализ требований к медицинским приложениям.
  + Источники: Юридические фирмы, законодательные базы.
* Инвесторы и партнеры:
  + Методы:
    - Сбор требований к ROI и маркетинговой стратегии.
    - Уточнение бюджетных ограничений.
  + Источники: Встречи с инвесторами, договоры с партнерами.

Потребители информации и способы доставки:

* Разработчики
  + - Ежедневные стендапы (Zoom/Teams).
  + - Jira/Confluence для управления задачами.
* Инвесторы
  + - Ежемесячные PDF-отчеты по email.
  + - Презентации через Zoom (каждые 2 месяца).
* Партнеры (маркетинг)
  + - Совещания в Teams.
  + - Общие документы в Google Drive.
* Пенсионеры (пользователи)
  + - Печатные брошюры (распространяются через соццентры).
  + - Видеоуроки на RuTube.
  + - SMS-оповещения.
* Семьи пенсионеров
  + - Email-рассылки.
  + - Push-уведомления в приложении.

Детальное описание документов:

1. Техническое задание (ТЗ)

* Формат: PDF/Docx (версии в Confluence/Jira).
* Содержание:
  + Функциональные требования (напоминания о лекарствах, мониторинг здоровья, аналитика).
  + Технические спецификации (совместимость с iOS/Android, офлайн-режим, интеграция с API).
  + Требования к безопасности (шифрование данных, аутентификация).
* Уровень детальности: Высокий (детализация для разработчиков и тестировщиков).
* Определения:
  + "Офлайн-режим" — базовые функции (напоминания, просмотр истории) доступны без интернета.
  + "API интеграция" — подключение к медицинским базам данных и сервисам аналитики.

2. Финансовый отчет

* Формат: PDF/XLSX (защищен паролем).
* Содержание:
  + Бюджет: расходы на разработку, маркетинг, тестирование.
  + ROI (возврат инвестиций): прогнозы и фактические показатели.
  + Риски: финансовые угрозы и меры их минимизации.
* Уровень детальности: Средний (ключевые метрики для инвесторов).
* Определения:
  + "ROI" — отношение чистой прибыли к сумме инвестиций.
  + "Критические риски" — задержки, превышение бюджета.

3. Маркетинговый план

* Формат: Google Docs/PPT.
* Содержание:
  + Целевая аудитория: пенсионеры и их семьи.
  + Каналы продвижения: соцсети, партнерские клиники, СМИ.
  + График запуска рекламных кампаний.
* Уровень детальности: Средний (стратегия + тактические шаги).
* Определения:
  + "KPI" — охват, конверсия, установки приложения.
  + "Пилотный запуск" — тестирование в ограниченной аудитории.

4. Руководство пользователя

* Формат: PDF/видео (с субтитрами).
* Содержание:
  + Пошаговая инструкция по установке и настройке приложения.
  + Раздел "Часто задаваемые вопросы" (FAQ).
  + Контакты поддержки (телефон, email).
* Уровень детальности: Низкий (простой язык, минимум технических терминов).
* Определения:
  + "Напоминание" — уведомление о приеме лекарств.
  + "Мониторинг здоровья" — ввод данных о давлении и сахаре.

5. Отчет о тестировании

* Формат: Excel/PDF (с графиками).
* Содержание:
  + Результаты тестирования функционала (баги, исправления).
  + Производительность: скорость работы, энергопотребление.
  + Рекомендации по доработке.
* Уровень детальности: Высокий (детали по каждому тест-кейсу).
* Определения:
  + "Критический баг" — ошибка, блокирующая работу приложения.
  + "Сценарий тестирования" — последовательность действий для проверки функции.

6. Политика конфиденциальности

* Формат: PDF (с электронной подписью).
* Содержание:
  + Типы собираемых данных (медицинские показатели, персональная информация).
  + Условия хранения и передачи данных.
  + Права пользователей (удаление аккаунта, доступ к данным).
* Уровень детальности: Средний (соответствие GDPR).
* Определения:
  + "Персональные данные" — ФИО, возраст, контакты.
  + "Анонимизация" — удаление идентифицирующей информации.

План ввода в действие выбранных видов коммуникаций

Этапы реализации:

1. Подготовительный этап (1-2 недели):
   * Настройка инструментов:
     + Создание аккаунтов в Jira/Confluence для управления задачами.
     + Настройка Google Workspace для совместной работы с документами.
     + Регистрация в Mailchimp/SendPulse для рассылок.
   * Обучение команды:
     + Проведение тренингов для разработчиков по работе с Jira.
     + Инструктаж маркетологов по использованию Mailchimp и Google Analytics.
   * Подготовка материалов:
     + Разработка шаблонов финансовых отчетов, руководств пользователя и политики конфиденциальности.
     + Создание видеоуроков для пенсионеров (с субтитрами).
2. Пилотный запуск (2 недели):
   * Тестирование коммуникаций:
     + Проведение пробных стендапов с разработчиками через Zoom.
     + Рассылка тестовых SMS-оповещений и пуш-уведомлений.
     + Распространение печатных брошюр в одном социальном центре для оценки удобства.
   * Сбор обратной связи:
     + Анкетирование участников пилота (пенсионеры, семьи, разработчики).
     + Анализ открываемости и кликабельности рассылок.
3. Полномасштабный запуск (4 недели):
   * Внедрение каналов:
     + Ежедневные стендапы для разработчиков (фиксация в Jira).
     + Регулярные финансовые отчеты для инвесторов (PDF + презентации).
     + Рассылка руководств пользователя через соццентры и RuTube.
     + Запуск чата поддержки в приложении и горячей линии.
   * Интеграция инструментов:
     + Настройка автоматических уведомлений в Firebase (push) и Twilio (SMS).
     + Подключение шифрования данных для передачи медицинской информации.
4. Мониторинг и корректировки (постоянно):
   * Метрики контроля:
     + Время реакции на запросы пользователей (цель — до 24 часов).
     + Процент доставки сообщений (SMS, email).
     + Удовлетворенность аудитории (опросы раз в квартал).
   * Корректирующие действия:
     + Оптимизация частоты рассылок при негативной обратной связи.
     + Дублирование информации через соцработников для пенсионеров без интернета.
     + Обновление глоссария терминов при выявлении недопонимания.

Методы обновления и совершенствования плана коммуникаций

1. Регулярный сбор и анализ обратной связи

* Методы:
  + Опросы пользователей: Ежеквартальные анкетирования пенсионеров и их семей через приложение, телефонные звонки или соццентры.
  + Фокус-группы: Обсуждение с представителями целевой аудитории для выявления проблем в коммуникации.
  + Аналитика данных: Мониторинг открываемости рассылок, кликабельности ссылок, времени реакции поддержки.
* Действия:
  + Корректировка каналов доставки (например, замена SMS на голосовые сообщения при низкой вовлеченности).
  + Обновление инструкций и видеоуроков на основе частых вопросов.

2. Адаптация к технологическим изменениям

* Методы:
  + Оценка эффективности текущих платформ (Jira, Mailchimp и др.) и поиск альтернатив.
  + Внедрение автоматизации: Использование AI-чатов для ответов на типовые запросы пользователей.
  + Интеграция новых каналов: Например, подключение мессенджеров (WhatsApp, Viber) для пенсионеров, активно использующих смартфоны.
* Действия:
  + Тестирование новых инструментов на ограниченной аудитории перед полным внедрением.
  + Обновление технической документации при изменении процессов.

3. Обучение и развитие команды

* Методы:
  + Тренинги: Ежегодные семинары по новым инструментам коммуникации (например, работа с ChatGPT для поддержки).
  + Внутренние вебинары: Обмен опытом между отделами (разработка, маркетинг, поддержка).
  + Ролевые игры: Моделирование сложных сценариев коммуникации (например, экстренные уведомления).
* Действия:
  + Создание базы знаний с инструкциями и лучшими практиками.
  + Введение системы наставничества для новых сотрудников.

4. Мониторинг законодательных изменений

* Методы:
  + Юридический аудит: Ежегодная проверка соответствия GDPR и локальных законов о данных.
  + Консультации с экспертами: Привлечение юристов для анализа новых требований.
* Действия:
  + Обновление политики конфиденциальности и пользовательских соглашений.
  + Информирование пользователей о изменениях через email и push-уведомления.

**План управления конфигурацией и версиями проекта**

**Введение**  
План управления конфигурацией и версиями проекта «Мобильное приложение для пенсионеров» разработан для обеспечения контроля над изменениями в программном коде, документации и процессах. Это позволит:

* Сохранять стабильность и работоспособность приложения на всех этапах жизненного цикла.
* Минимизировать риски конфликтов версий и ошибок, вызванных несогласованными изменениями.
* Обеспечить прозрачность и отслеживаемость всех модификаций.

Важность управления конфигурацией для проекта:

* Для разработки:
  + Упрощение совместной работы команды над кодом (например, через ветвление в Git).
  + Автоматизация тестирования и развертывания (CI/CD).
* Для пользователей:
  + Гарантия безопасности и надежности обновлений.
  + Возможность отката к предыдущим версиям при критических сбоях.
* Для бизнеса:
  + Соблюдение сроков релизов и бюджетных ограничений.
  + Соответствие требованиям GDPR и защита персональных данных.

Связь с другими документами проекта:

* План коммуникаций (Jira/Confluence для управления задачами).
* Политика конфиденциальности (хранение и передача данных).
* Отчеты о тестировании (версии приложения и исправления багов).

Основные термины:

* Конфигурационная единица: Любой компонент проекта, подлежащий контролю (код, документация, настройки серверов).
* Версионирование: Присвоение уникальных идентификаторов (например, v1.2.3) для отслеживания изменений.
* Релиз: Стабильная версия приложения, готовая для публикации в магазинах.

Используемые инструменты:

* Git — для управления версиями кода.
* Jira/Confluence — для отслеживания задач и документации.
* GitHub Actions — для автоматизации CI/CD.
* Docker — для контейнеризации зависимостей.

Ожидаемые результаты:

* Четкий процесс внесения и утверждения изменений.
* Сокращение времени на устранение конфликтов в коде.
* Повышение доверия пользователей за счет стабильных обновлений.

**Конфигурационное управление программным продуктом**

Технические аспекты

1. Инструменты управления версиями и конфигурацией:
   * Git (GitHub):
     + Стратегия ветвления: Используется адаптированный Git Flow.
       - main — стабильная версия, соответствующая релизам.
       - develop — ветка для интеграции новых функций.
       - feature/\* — разработка новых функций.
       - hotfix/\* — экстренные исправления для main.
     + Тегирование версий: Семантическое версионирование (SemVer) — v1.2.3, где:
       - 1 — мажорная версия (критические изменения),
       - 2 — минорная версия (новые функции),
       - 3 — патч (исправления багов).
   * Jira + Confluence:
     + Связь задач Jira с коммитами (например, JIRA-123: Добавлен мониторинг давления).
     + Хранение технической документации в Confluence с версионированием.
   * Docker:
     + Контейнеризация зависимостей для единообразия сред (разработка, тестирование, продакшен).
2. CI/CD (GitHub Actions):
   * Этапы пайплайна:
     1. Сборка: Компиляция кода и создание артефактов.
     2. Тестирование:
        + Юнит-тесты, интеграционные тесты.
        + Проверка безопасности (статические анализаторы, например, SonarQube).
     3. Деплой:
        + Тестовое окружение: Автоматический деплой ветки develop на staging-сервер.
        + Продакшен: Ручное подтверждение деплоя из main через Pull Request.
   * Артефакты:
     1. Собранные APK/IPA-файлы хранятся в GitHub Releases.
     2. Docker-образы публикуются в приватный Docker Registry.
3. Хранение конфигураций:
   * Конфигурационные файлы (настройки БД, API-ключи):
     1. Шаблоны хранятся в Git с суффиксом .template (например, config.yaml.template).
     2. Реальные значения подставляются через переменные окружения или Secrets (GitHub Secrets).
   * Окружения:
     1. dev — локальные машины разработчиков.
     2. staging — тестовый сервер, идентичный продакшену.
     3. production — рабочий сервер.

Технологические аспекты

1. Управление зависимостями:
   * Мобильная разработка (iOS/Android):
     + Использование CocoaPods (iOS) и Gradle (Android) с фиксацией версий в Podfile.lock/build.gradle.
2. Совместимость и офлайн-режим:
   * Локальная БД: SQLite для хранения данных о лекарствах и показателях здоровья в офлайн-режиме.
   * Синхронизация: При восстановлении связи данные синхронизируются с облачной БД (Firebase).
3. Безопасность:
   * Шифрование:
     + Данные пользователей шифруются на устройстве (AES-256).
     + HTTPS для передачи данных в облако.
   * Резервные копии:
     + Ежедневные бэкапы конфигураций и БД в AWS S3.

Контроль изменений

1. Код-ревью:
   * Обязательное ревью Pull Request минимум двумя разработчиками.
   * Проверка на соответствие стандартам кода и покрытие тестами.
2. Согласование релизов:
   * Релиз-менеджер утверждает мердж в main после:
     + Успешного прохождения CI/CD.
     + Тестирования на staging.
     + Подписания чек-листа (например, проверка безопасности).
3. Откат изменений:
   * Автоматический откат через rollback-деплой в GitHub Actions (предыдущий Docker-образ).
   * Ручной откат к тегу предыдущей версии (например, git checkout v1.2.2).

**Программа конфигурационного управления**

Политики и процедуры

1. Управление изменениями:
   * Подача запроса на изменение:
     + Все изменения (код, документация, настройки) регистрируются в Jira как задачи с типом «Improvement» или «Bug».
     + Запрос должен включать описание, обоснование и оценку влияния на другие компоненты.
   * Утверждение изменений:
     + Изменения в код утверждаются через Pull Request (PR) в GitHub.
     + Для критических изменений (например, обновление ядра приложения) требуется согласование с архитектором проекта.
2. Версионирование:
   * Код: Семантическое версионирование (SemVer) для релизов.
   * Документация: Версии в Confluence помечаются датой и номером (например, v1.0-2023-10-05).
3. Резервное копирование:
   * Ежедневные бэкапы репозиториев, БД и конфигураций в AWS S3.
   * Хранение резервных копий не менее 30 дней.

Критерии приемки изменений

1. Для кода:
   * Успешное прохождение CI/CD (сборка + тесты).
   * Покрытие кода тестами не менее 80% для новых функций.
2. Для документации:
   * Соответствие шаблонам в Confluence.
   * Согласование с техническим писателем.
3. Для релизов:
   * Тестирование на staging не менее 48 часов.
   * Подписание чек-листа релиза (безопасность, производительность, UX).

Управление документацией

* Конфигурационные файлы:
  + Хранятся в Git в папке configs.
* Схемы БД и API:
  + Версионируются в Swagger/OpenAPI и диаграммах ERD (хранятся в Confluence).
* Журнал изменений (Changelog):
  + Ведется автоматически через GitHub Releases на основе тегов и PR.

Взаимодействие с другими процессами

1. Связь с задачами в Jira:
   * Каждый коммит ссылается на задачу в формате JIRA-123.
   * Закрытые задачи автоматически обновляют статус в Jira через GitHub Actions.
2. Интеграция с тестированием:
   * Падение тестов в CI/CD блокирует мердж в main.
   * Отчеты о тестировании прикрепляются к PR.
3. Связь с планом коммуникаций:
   * Уведомления о релизах рассылаются через email и push-сообщения.
   * Изменения в конфигурациях согласуются с командами поддержки и маркетинга.

Экстренные сценарии

* Критический баг в продакшене:
  1. Создать ветку hotfix/\* от main.
  2. Исправить ошибку, протестировать на staging.
  3. Мердж в main и develop с тегом патча (например, v1.2.4).
  4. Автоматический откат через CI/CD, если фикс не сработал.
* Потеря данных:
  1. Восстановление из последнего бэкапа в AWS S3.

**Этапы управления конфигурацией**

1. Идентификация конфигурационных единиц

* Что включает:
  + Код приложения: Модули iOS (Swift), Android (Kotlin), бэкенд.
  + Документация: Техническое задание, руководства пользователя, политика конфиденциальности.
  + Настройки: Конфигурационные файлы (API-ключи, настройки БД), Docker-образы.
  + Артефакты: Собранные APK/IPA-файлы, тестовые отчеты.
* Методы идентификации:
  + Уникальные идентификаторы для каждой единицы (например, app-android-v1.3.0, doc-privacy-policy-2023).
  + Каталогизация в Confluence с указанием версий и зависимостей.

2. Контроль изменений

* Процедура:
  1. Создание задачи в Jira: Описание изменения, его цели и влияния.
  2. Разработка в изолированной ветке: Например, feature/add-medication-reminder.
  3. Код-ревью: Проверка PR минимум двумя разработчиками.
  4. Интеграция в develop: После успешного CI/CD и тестирования.
  5. Релиз: Мердж develop в main с тегом версии (например, v1.4.0).
* Исключения:
  1. Hotfix: Изменения вносятся напрямую в main через ветку hotfix/\*, затем синхронизируются с develop.

3. Учет статусов конфигураций

* Статусы:

| Статус | Описание |
| --- | --- |
| Черновик | Изменение в разработке (ветка feature/\*). |
| На ревью | PR создан, ожидает проверки. |
| Протестировано | Успешное прохождение CI/CD и тестов. |
| В продакшене | Версия активна в main и доступна пользователям. |

* Инструменты:
  + Jira для отслеживания статусов задач.
  + GitHub Projects для визуализации этапов.

4. Аудит конфигурации

* Цели:
  + Проверка соответствия кода и документации требованиям.
  + Выявление неучтенных изменений.
* Периодичность:
  + Перед каждым релизом — полный аудит.
  + Ежеквартально — выборочная проверка.
* Методы:
  + Сравнение версий в Git с журналом изменений в Jira.
  + Проверка безопасности через SonarQube и ручное тестирование.

5. Релиз и развертывание

* Процесс:
  1. Подготовка:
     + Создание тега версии в Git (v1.4.0).
     + Сборка артефактов через GitHub Actions.
  2. Тестирование на staging:
     + Проверка всех функций и интеграций.
     + Мониторинг производительности (например, нагрузка на сервер).
  3. Публикация:
     + Загрузка APK/IPA в App Store и Google Play.
     + Обновление Docker-образов на продакшен-серверах.
  4. Уведомление:
     + Рассылка релиз-нот через email и push-сообщения.
     + Обновление Changelog в GitHub Releases.

6. Откат изменений

* Сценарии:
  + Критический баг: Возврат к предыдущей версии через тег (например, v1.3.2).
  + Нестабильность продакшена: Автоматический откат через CI/CD (используется предыдущий Docker-образ).
* Документирование:
  + Причина отката фиксируется в Jira.
  + Информация включается в отчёт о релизе.

**Обучение и ресурсы**

Обучение команды

1. Для разработчиков:
   * Git и GitHub:
     + Базовый курс: ветвление, мердж, разрешение конфликтов.
     + Продвинутый: работа с Git Flow, семантическое версионирование.
   * CI/CD (GitHub Actions):
     + Настройка пайплайнов, интеграция тестов, деплой на staging/production.
   * Инструменты безопасности:
     + Использование SonarQube для статического анализа кода.
   * Формат:
     + Онлайн-курсы (платформы типа Udemy).
     + Внутренние воркшопы от техлида.
2. Для тестировщиков:
   * Автоматизированное тестирование:
     + Написание скриптов на Python/JavaScript для интеграционных тестов.
   * Работа с Jira:
     + Создание тест-кейсов, привязка к задачам, отчеты о багах.
   * Формат:
     + Практические сессии с ментором.
3. Для менеджеров и релиз-менеджеров:
   * Управление версиями:
     + Понимание жизненного цикла релизов, работа с Changelog.
   * Инструменты:
     + Основы Jira для отслеживания прогресса.
   * Формат:
     + Вебинары от администратора DevOps.
4. Для новых участников:
   * Онбординг:
     + Стартовый пакет: доступ к репозиториям, шаблоны документов.
     + Назначение ментора из опытных разработчиков.
   * Ресурсы:
     + Видеоуроки по настройке локального окружения.

Необходимые ресурсы

1. Технические ресурсы:
   * Репозитории:
     + GitHub (приватные репозитории для кода).
     + Docker Hub (хранение образов).
   * Инструменты CI/CD:
     + GitHub Actions (бесплатный для публичных репозиториев, корпоративная лицензия для приватных).
   * Инфраструктура:
     + Серверы для staging (AWS EC2).
     + Базы данных (Firebase, MongoDB Atlas).
2. Документация:
   * Руководства:
     + «Как работать с Git Flow» (Confluence).
     + «Настройка CI/CD с нуля» (PDF + видео).
   * Шаблоны:
     + PR-шаблон с чек-листом для код-ревью.
     + Шаблон тест-кейсов в Jira.
3. Лицензии и доступы:
   * ПО:
     + JetBrains IDE (лицензии для разработчиков).
     + SonarQube (корпоративная подписка).
   * Облачные сервисы:
     + AWS (бюджет выделяется на хранение и серверы).
     + Google Workspace (для совместной работы).

Поддержка и обновление знаний

1. Регулярные обновления:
   * Ежеквартальные тренинги по новым инструментам (например, переход на GitHub Copilot).
   * Рассылка дайджестов с обновлениями в CI/CD или безопасности.
2. База знаний:
   * Раздел в Confluence с часто задаваемыми вопросами (FAQ).
   * Чат-бот в Slack для быстрого поиска документации.
3. Мониторинг потребностей:
   * Ежегодный опрос команды на тему «Какие навыки требуют улучшения?».
   * Введение системы грантов на курсы (например, Coursera).