Разработка мобильных приложений

Лекция 1

Введение и практическое задание

Ключев А.О. к.т.н., доцент ФПИиКТ Университета ИТМО

Санкт-Петербург

2026

Контакты

- Ключев Аркадий Олегович
- E-mail: kluchev@yandex.ru
- ВК: https://vk.com/aokluchev
- Группа “РМП 2026 весна” в Telegram

Литература

На этом слайде представлены универсальные книги содержащие фундаментальные концепции, которые живут десятки лет. Концепции из этих книг пригодятся вам практически в любой области IT-технологий. По каждой отдельной теме будет специальная литература.

- Роберт Мартин. Чистая архитектура.
- Фредерик Брукс. Мифический человеко-месяц, или как создаются программные системы.
- Непейвода Н. Н. ,Скопин И. Н. Основания программирования.
- Гради Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++

- Кстати, почему с книгами проблема и не имеет смысла покупать все книги подряд?

Литература по управлению проектами (для рабочих групп)

- Джеф Сазерленд. Scrum. Революционный метод управления проектами.
- Скрам. Описание
- Скрам Гайд. Исчерпывающее руководство по Скраму: Правила Игры
- Майк Кон. SCRUM. Гибкая разработка ПО.

В чем проблема университетов?

- Я посмотрел много роликов на YouTube, где успешные выпускники ругают свои университеты и вспомнил время, когда я был студентом. Вот основные претензии:
  - Старые, равнодушные и ленивые преподаватели, не желающие осваивать новые технологии
  - Заносчивые и высокомерные преподаватели, унижающие и оскорбляющие своих студентов
  - Некомпетентные преподаватели, которые на самом деле ничего не знают и не умеют, если их взять на работу
  - Устаревшее оборудование/программное обеспечение
  - Странные учебные программы, не совпадающие с реальностью индустрии IT технологий
  - Никому не нужные знания, отстающие от того, что используется в реальных проектах на пару десятков лет
    - Например, я изучал языки Pascal и Fortran, которым я ни разу в жизни не пользовался, а также ассемблер ЕС ЭВМ, от которого тоже не было никакого толка. Я изучал теорию, которую никто не мог применить на практике, потому что не знал как. Мне нужны были знания по архитектуре компьютеров и контроллеров, сетям, программной инженерии, современным ОС (я хотел заниматься Unix), языкам типа C/C++ и по микроконтроллерам, но этого не было в программе.
    - То, что мне реально требовалось на работе (на кафедре (!), а потом в фирме занимающейся разработкой промышленных контроллеров) я изучал сам, правда не без помощи хороших преподавателей кафедры, имеющих богатый практический опыт.
  - В результате ~~я прогулял большинство занятий~~ я закончил университет с хорошими оценками
- Вам нужно научиться как-то обходить подобные проблемы (если они появятся) и получать пользу от занятий, чтобы не терять свое время зря! **Помните, что не вас учат, а Вы учитесь!**

Что такое университет?

- Это фильтр. Вы обратили внимание, что вокруг вас в основном умные люди? Поступить в ИТМО довольно непросто.
- Концентрация специалистов. Университет это место, где сконцентрировано много интересных людей, обладающих разными знаниями, умениями, характерами, особенностями…
- Социализация. Вы привыкаете работать в коллективе. Вас готовят к тому, чтобы вы потом работали в корпорации или создали свой собственный бизнес.
- Полезные связи. Вы можете познакомится со своими будущими коллегами или работодателями.
- Разные подходы и разные стили обучения. Примерно так работает N-версионное программирование. У каждого преподавателя свои способы до вас достучаться. Даже если какой-то курс вам сразу не понравился, вы все равно можете извлечь из этого для себя что-то полезное.
- Это путь с полосой препятствий. В процессе прохождения этого пути вы совершенствуете свой мозг, а диплом в конце обучения это доказательство полученных вами компетенций.
- Это короткий путь. Разные специалисты, представители бизнеса, индустрии и ваши друзья показывают вам как пройти путь обучения как можно быстрее и эффективнее.

Как учиться?

- Обучение = тренировка головного мозга
- По мере обучения вы совершенствуете свой мозг и решать профильные задачи становится все проще и проще. Чем больше практики, тем проще.
- Обучение в университете чем-то похоже на тренировку в спортзале. Только в одном случае вы тренируете свой мозг, а в другом тело и мышцы.
- Можно ли стать бодибилдером не занимаясь тяжестями, а просто сидя на диване и слушая лекции по бодибилдингу? Нет. Точно также нельзя стать инженером-программистом, слушая одни лекции по программированию. Вы должны тренировать свой мозг, а для этого вам нужно максимально задействовать свой мозг:
  - Много читать, смотреть видеоролики по профильным темам и **думать**
  - Решать множество технических задач, постепенно увеличивая их сложность.
- Мир технологий постоянно меняется, поэтому учиться вам придется всю жизнь.

О ценности знаний

В 358 аудитории есть полка с книгами, посмотрите на эти книги и год их выпуска, когда будете на лабах на Биржевой. Это поучительно.

Как закончить университет без какого-либо смысла

- Везде проскакивать по пути минимального сопротивления
- Не общаться с окружением
- Не участвовать в проектах
- Не делать докладов на конференциях и не писать статей
- Не участвовать в НИР
- Не работать по специальности
- Не читать дополнительную литературу
- Ничего не изучать по специальности
- Не развивать свой мозг
- …
- Профит! Возможно вы получите свои тройки и получите корочку о высшем образовании, но как специалист вы будете так себе… Я бы такого не взял себе на работу.

Проблемы IT технологий

- Технологии быстро меняются. Как в таких условиях держать руку на пульсе современных IT технологий и быть востребованным на рынке труда?
- Стеки технологий создаются не великими учеными, а обычными программистами (со всеми вытекающими проблемами). Развитие технологий крайне многовекторно, идет эволюционным путем и оно довольно бессистемно. Победу той или иной технологии определяет успех на рынке, а не техническое совершенство и перспективы в будущем (например, см. историю взаимоотношений Н. Вирта и фирмы Borland).
- Как не потерять работу из-за конкуренции с ИИ в ближайшем будущем?
- Какие языки программирования и стеки технологий учить?
- Какие книги читать?

Что изменил ИИ?

- Теперь не имеют особой ценности следующие навыки:
  - Умение искать информацию – нейросеть справляется быстрее и лучше
  - Умение программировать (кодировать) на разных языках
  - Знание рутинных и простых операций (например, создание простых программ, установка программ и их настройка, создание тестов и т.п.). ИИ-агенты выросли до уровня джуниоров/мидлов.
- Кардинально изменился подход к программированию, производительность специалистов выросла многократно
- Ценность теперь имеют тимлиды, глубоко понимающие архитектуру, умеющие ставить задачи и проверять результат.

- Конец привычного программирования https://www.oreilly.com/radar/the-end-of-programming-as-we-know-it/

Университет или курсы?

- Многие сейчас говорят, что в области IT технологий высшее образование не имеет никакого смысла
- Университет готовит инженеров, умеющих самостоятельно решать широкий спектр задач на базе имеющихся знаний или ученых (магистров, кандидатов и докторов наук), умеющих создавать новые знания.
- Курсы готовят «жрецов», умеющих пользоваться набором заклинаний, но не понимающих как они устроены (см. цикл романов (не сериал, а именно книги!) «Основание» Азимова).
- Если вам нужен специалист, с перспективой роста и работой на годы, то лучше брать на работу инженера. Если вам нужно сиюминутно заткнуть дыру в проекте с какой-то не особо интеллектуальной позицией, то подойдет «жрец» после курсов. Прелесть «жрецов» в том, что они легко взаимозаменяемы.
- Тем не менее курсы полезны (особенно бесплатные), так как они позволят вам быстрее понять как сделать лабораторные работы.

Цель курса

- Показать подход к изучению нового стека технологий на примере экосистемы Android. Для этого мы собираемся:
  - Изучить основные концепции экосистемы Android (это базовые концепции, которые кочуют из проекта в проект годами и десятками лет без каких либо серьезных изменений).
  - Зная концепции, научиться пользоваться текущим стеком технологий экосистемы Android. Для этого необходимо научиться видеть концепции в описании стека технологий (это довольно непросто, так как устоявшейся терминологии как правило нет).
- Итак, мы хотим найти способ решения целого спектра разных технических задач в области вычислительной техники, требующих изучения неизвестного стека технологий на примере Android.

В чем состоит специфика обычных курсов по Andoid?

- Вас готовят к встраиванию в имеющийся производственный процесс (создание приложений на базе текущей версии Android), как правило для самых простых градаций (Junior).
- Вас учат решать шаблонные задачи
- Вам объясняют как решать определенный круг задач, но не объясняют почему именно так нужно решать задачи и как это все устроено внутри. Решая задачу вы скорее всего не сможете объяснить никому что вы делаете. Действие напоминает заклинания из волшебной книги, которые вы запомнили, но не можете объяснить. Еще этот процесс можно сравнить с китайской комнатой.
- При появлении нового стека технологий вам придется снова идти на курсы.

В чем ваше преимущество как студентов?

- В вашей рабочей группе командуете вы, вы никому ничего не должны и ваш проект ограничивается только вашей фантазией, семестром и физическими возможностями
- Вы можете посмотреть на разные подходы к программированию, различные варианты архитектуры и изучить то, что вам кажется полезным и интересным
- При желании можно расширить рамки проекта

Курс «разработка мобильных приложений»

- Групповое задание – приложения для Android + бэкэнд (группа до 10 человек)
- Что сдаем:
  - Отчет в формате PDF должен содержать 4 основных раздела + оглавление, введение, заключение и список литературы:
    1. Техническое задание (делает ИИ агент по вашим требованиям)
    2. Описание архитектуры (**делаем сами, это самое важное!!!**)
    3. Описание реализации (делает ИИ агент)
    4. Описание системы тестирования (делает ИИ агент)
  - База знаний в Obsidian (это часть проекта) – создается из статей полученных от ИИ в процессе понимания матчасти
  - Исходные тексты проекта (zip архив из репозитория git вместе с базой знаний)
  - Документация к проекту (.md, создается ИИ агентом)
  - Презентация проекта
- Для выполнения лабораторных работ желательно иметь с собой ноутбук с Android Studio
- Если нет подходящего телефона с Android можно использовать симулятор.

Про лекции в курсе

- В настоящий момент я не зарабатываю денег на проектировании мобильных приложений и я не буду изображать глубокого специалиста.
- Я не буду углубляться в тонкости конкретных версий Android по целому ряду причин:
  - Объем этих знаний огромен, он не уложится в короткий курс;
  - Я этими знаниями не владею в должной мере (я всегда разбирался с тонкостями в рабочем порядке, а потом благополучно забывал все, так как голова не резиновая!);
  - Изучать тонкости и лезть слишком глубоко в технологии впрок нет смысла, так как пока вы учитесь эти знания перестанут быть актуальными и без практики вы скорее всего все быстро забудете.
- **Но как нам тогда быть?**
- **Наша задача проста. На примере данной вам задачи вы должны выработать навык получения новых знаний о конкретных технологиях и научиться быстро этими технологиями пользоваться на практике.**

Основной акцент курса – учимся учиться

- Мы не столько делаем приложение, сколько пытаемся понять как это делать и изучаем стек технологий
- Мы используем научный подход
- Мы пытаемся записывать факты в базу знаний на основе Obsidian и связывать их в систему
- В конце курса пытаемся рассказать про проект

Как мы делаем проект?

- Мы пишем код с помощью ИИ агентов (Qwen Code, Roo Code, Open Code, …)
  - Мы должны уметь использовать ИИ агенты
  - Мы должны уметь ловить ИИ «за руку», когда он несет чушь
  - Мы должны уметь создавать и **понимать архитектуру проекта**
  - Мы должны уметь дописывать куски кода руками (обязательно!)
  - Мы должны уметь тестировать свой проект
  - Мы должны уметь заставить ИИ сделать удобоваримую документацию к проекту

Контрольные работы

- У нас по плану 5 контрольных, но они будут на каждой лекции (до 8 штук) для мотивации посещения лекций и закрепления материала предыдущей лекции
- Тема контрольной берется из предыдущей лекции, как часть вашего отчета по лабораторной работе или по всему пройденному материалу (если контрольная рубежная).
- Обычно 3 вопроса
- Ответ в свободной форме
- В чем смысл контрольных? Когда вы пишете – вы думаете, а думать полезно.

Что мы вообще делаем?

- Мы имитируем небольшой стартап и пытаемся сделать минимально жизнеспособный вариант приложения (MVP) с документацией за этот семестр.
- Я понимаю, что у вас куча других предметов, вы работаете, устали и вообще у вас лапки. Поэтому программы могут быть простыми, им разрешается иногда падать, а в документации допускаются пробелы.
- Но, вам предстоит работать и я надеюсь, что в этом курсе вы узнаете что-то полезное для себя.

Выбор операционной системы для работы

- Желательно иметь на своем ноутбуке **Linux** (единственной или второй системой). Я предпочитаю Ubuntu или Linux Mint. По моему опыту с Linux меньше всего проблем. В мире разработчиков софта популярны Mac OSx и Linux
- Linux удобнее, если вам потребуется запускать какие-то серверные приложения. Большинство серверов и СУБД ставятся одним кликом из репозитория. Более того, собрать Android из исходников можно только в Linux (вдруг кому-то захочется получить свой Andoid).
- В Mac OSx это делать обычно чуть сложнее чем в Linux, зато эргономика Macbook почти идеальна и никто не мешает использовать docker контейнеры.
- Windows лично мне не нравится и я его практически не использую. В Windows я запускаю Linux в виртуальной машине, использую docker контейнеры или Windows System for Linux (WSL, WSL2). WSL/WSL2 работает на Windows 10+

Стек технологий

- IDE
  - InelliJ IDEA и ее клоны
    - Android Studio
    - PyCharm (Python)/JupyterLab
    - OpenIDE (есть свой репозиторий плагинов в РФ), GigaIDE, InteliJ IDEA CE (не ставятся плагины)
  - Visual Studio Code (наверно лучший вариант, хорошо подключаются ИИ агенты)
  - Zed (довольно убогий встроенный ИИ агент)
- ИИ чаты, агенты
  - Qwen code, DeepSeek – **пока** бесплатно
  - Любые варианты если вы хотите и можете платить за доступ к API
- Языки программирования
  - Android - Kotlin
  - бэкэнд – Kotlin, Go, C
- Брокер сообщений для бэкенд - Redis Server/KeyDB (самый простой вариант), но на самом деле можете использовать что угодно
- СУБД для бэкенд – PostgreSQL или аналоги
- Логи – Open Telemetry
- Хранение больших объемов данных и быстрая выдача клиенту – Clickhouse DB https://clickhouse.com/docs/ru
- Система контроля версий - Git (GutHub, GitLab,GitFlic, GitVerse и т.п.)
- Документация в исходных текстах (Dokka) + Markdown + Obsidian
- Отчет - PDF (с оглавлением, введением, заключением, списком литературы и красивыми картинками, оформленный по ГОСТ, почти как диплом)

Тема проекта – ПО для трейдинга и инвестиций

- В этом семестре вы должны разработать экосистему для поддержания трейдинга и инвестиций
- Группы до 10 человек, делается приложение на Android и бэкенд

Зачем нужна экосистема для трейдинга?

- Имитация бэкенда брокера, который получает информацию о котировках акций с «биржи»
- Имитация брокерского терминала для клиентов брокера (Android)
- Система поддерживает 10к клиентов

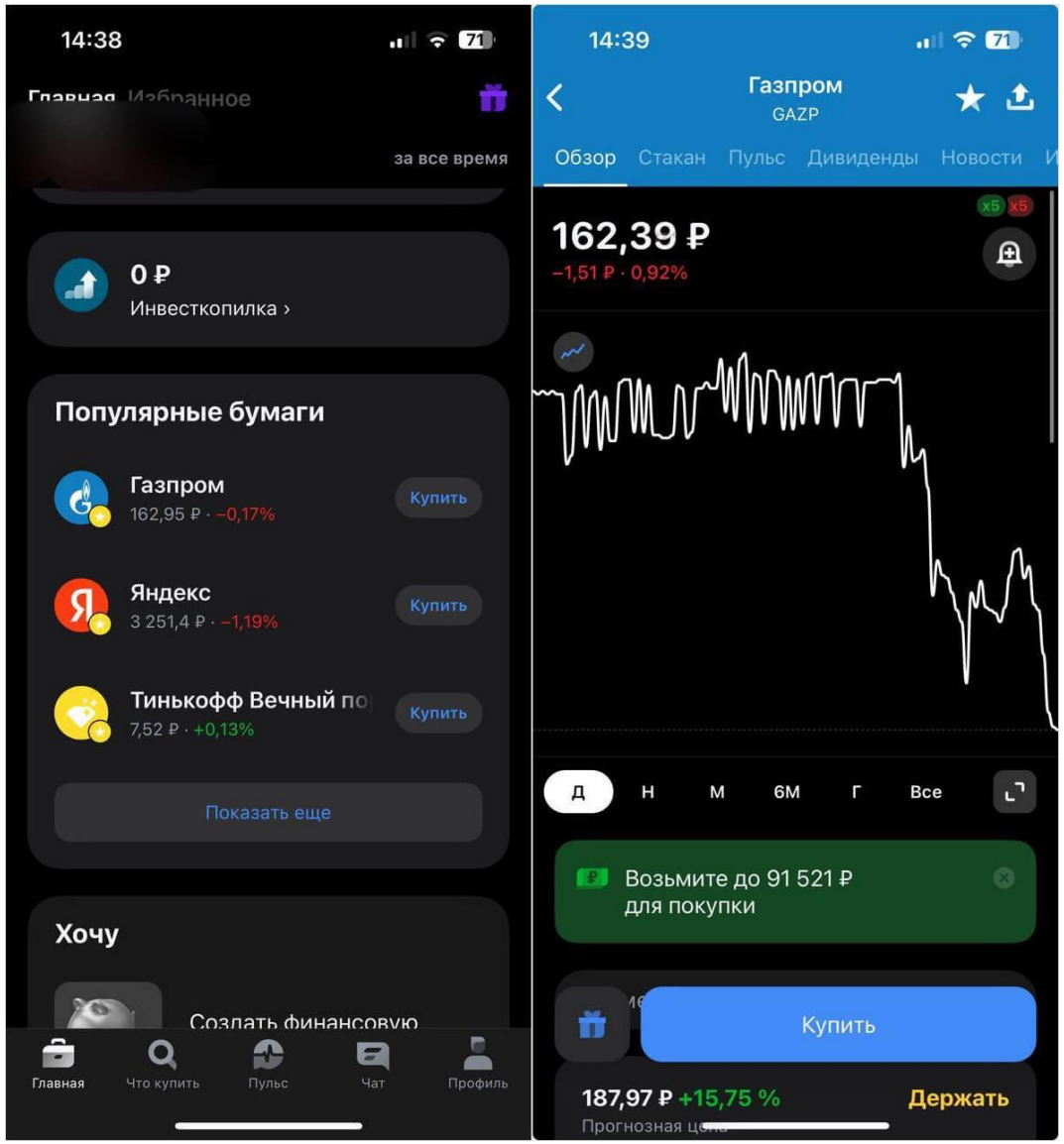
Стек технологий

- Нативное мобильное приложение:
  - **Android** (реальное устройство или симулятор)
  - **Kotlin**
  - **Jetpack compose**
- Мобильное приложение на базе Web фреймворка (React native и т.п.)
- Бэкенд
  - *nux: Linux (Windows/WSL), MacOS
  - Kotlin, Ktor, корутины, библиотека для сериализации JSON и т.п.
  - Go для сбора данных о котировках
  - C для драйвера Linux
  - Redis Server/KeyDB – кэш БД, брокер сообщений
  - PostgreSQL и аналоги (голый SQL)
  - ClickHouse
  - Open Telemetry – для сбора телеметрии
- За отклонение от этого списка буду снижать балл (все как в реальных проектах, только в реальных проектах вы просто не сдадите работу заказчику, если ваша работа не будет соответствовать техническому заданию)
- Проблема в том, что вас много, а всеми технологиями одновременно я к сожалению не владею… =(

Требования к экосистеме

- Создание системы для обслуживания 10-ти тысяч клиентов
- Состав системы:
    1. Мобильное приложение для Android (Kotlin + Jetpack Compose)
    2. Кросс-платформенное мобильное приложение (React Native и подобные)
    3. Микросервис для связи с мобильными приложениями (Ktor + Kotlin)
    4. Микросервис для работы с БД (Kotlin)
    5. Микросервис имитатор 10-ти тысяч мобильных приложений для тестирования системы (без GUI, просто рандомные запросы через API к серверу) – Kotlin, Python, Go,…
    6. Микросервис для получения котировок – Go
    7. Драйвер для Linux, с имитацией котировок - C
- Функции системы
    - Получение данных от биржи
    - Выдача данных клиентам
    - Организация покупки/продажи акций в терминале

Пример системы – Тинькофф инвестиции



Зачем нужна рабочая группа?

- Вам дается попытка сделать свой проект в коллективе (это непросто)
- Сложность проекта можно увеличить до размера, похожего на реальный проект
- Сложность проекта достаточно велика, чтобы в нем появилась необходимость в грамотном проектировании кода и в архитектуре.

Почему в рамках курса по разработке мобильных приложений будет какой-то бэкенд?

- В настоящее время оторванных от сети Интернет мобильных приложений очень мало и это что-то простое, типа калькулятора
- Мобильная разработка не очень сложна, как правило весь функционал расположен в облаке, а в мобильном приложении есть только интерфейс пользователя
- Мобильные технологии очень быстро меняются, нет смысла углубляться в тонкости, так как через пару лет все будет иначе.

Зачем нужен отчет?

- На самом деле это не совсем отчет, это так называемая «пояснительная записка», то есть документ являющийся описанием проекта.
- Практика показывает, что большинство студентов (по крайней мере среди тех, кого я брал на работу) не умеет писать документацию, что снижает качество проектов и замедляет карьерный рост сотрудников. Умение формулировать мысли говорит о том, что у человека порядок в голове и косвенно подтверждает гипотезу о том, что в проекте у такого человека тоже все более-менее нормально и он понимает, что он делает.

Оформление отчета

- Титульный лист
- Оглавление
- Введение

1. Техническое задание
2. Архитектура системы
3. Описание реализации
4. Тестирование

- Заключение
- Список литературы
- Приложения

Оформление текста

- Как в дипломе, пока нет особо жестких требований. Скачайте методичку про оформление диплома, например эту:
  - Калинина М.И., Смирнов С.Б. Методические указания по выполнению выпускных квалификационных работ для бакалавров - Санкт-Петербург: Университет ИТМО, 2016. - 40 с. - экз.
- Государственные стандарты (ГОСТы). Посмотрите, лучше привыкнуть к правильному оформлению документации сразу:
  - ГОСТ 2.105-95 «Единая конструкторская документация. Общие требования к текстовым документам».
  - ГОСТ 7.32-2017 «Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления».

Титульный лист

- Название проекта
- ФИО авторов
- Руководитель
- Организация (в нашем случае Университет ИТМО)
- Год

- Есть стандартные формы титульного листа

Оглавление

- Зачем нужно? Оглавление позволяет быстро оценить структуру документа и прочитать только те разделы, которые реально полезны в работе.
- Стоимость создания оглавления низка, а польза очень высока.
- Для создания оглавления можно воспользоваться любым вордо-подобным редактором (Мой офис, Яндекс 360 и т.п.) и создать там оглавление. Оглавление создается автоматически, если вы будете использовать в названиях разделов стили «Заголовок 1», «Заголовок 2» и т.д.
- Количество уровней заголовков лучше не делать больше трех, так как большее количество уровней плохо воспринимается при чтении таких маленьких документов.

Колонтитулы и номер страниц

- Внизу страницы (за исключением титульного листа) нужно вставить номер страницы. Исключительно полезная штука, если у вас документ распечатан и вам нужно найти нужный раздел. В чисто электронном документе есть гиперссылки, но все равно номера страниц крайне полезны.

Введение

- Введение дает читателю возможность быстро разобраться о назначении данного документа. Нужно написать:
  - Актуальность работы
  - Цели и задачи работы
- Актуальность – аргументы доказывающие важность, нужность, своевременность и необходимость данной разработки.
- Цель работы – получение чего-либо полезного, ответ на вопрос «ЗАЧЕМ?». Например, наша цель - согреться. Задачи – последовательность действий для достижения нашей цели. В случае с целью «согреться», нам нужно решить следующие задачи:
  - Найти сухие ветки
  - Сложить костер
  - Найти спички, зажигалку или каким либо иным способом получить огонь.
  - Разжечь костер.

Техническое задание

- Техническое задание (ТЗ) — перечень требований, условий, целей, задач, **поставленных заказчиком** (*) в письменном виде, документально оформленных и выданных исполнителю работ проектно-исследовательского характера. Такое задание обычно предшествует разработке проектов и призвано ориентировать разработчика на создание проекта, удовлетворяющего желаниям заказчика и соответствующего условиям использования, применения разрабатываемого проекта, а также ресурсным ограничениям.
- *(*) На самом деле ТЗ пишет исполнитель в 99% случаев*
- За основу можно взять ГОСТ 19.201-78 «Техническое задание, требования к содержанию и оформлению»

Разделы ТЗ в вашем отчете

- Введение;
- Назначение разработки;
- Требования к программе или программному изделию;
  - Функции системы
  - Состав системы
  - Характеристики системы

Как писать техническое задание?

- Составляете список заинтересованных лиц (например, мы делаем какую-то систему для какого-то завода)
  - Директор завода
  - Мастер цеха
  - Бухгалтер
  - Главный инженер
  - Рабочий
  - Охранник
  - Дворник
  - Гость/посетитель
  - Разработчик системы
- Составляете список важных аспектов разработки (инструментальный, прикладной, надежность, безопасность, энергопотребление (мобильная разработка!) и т.п.)
- Составляете список требований для каждого из заинтересованных лиц
- Оцениваете сроки, бюджет разработки и возможности разработчиков
- Формулируете ТЗ, которое вы можете успешно выполнить и которое удовлетворит заказчика.

Архитектура проекта

- Архитектура проекта позволяет понять как устроен проект внутри, из каких частей состоит, как эти части друг с другом связаны и как они друг с другом взаимодействуют.
- Для кого нужна архитектура?
  - Для технических руководителей.
  - Для архитекторов проектов.
  - Для старших разработчиков.
- Насколько глубоко нужно погружаться в архитектуру? Погружаться нужно до уровня, достаточного для создания спецификаций для загрузки работой рядовых программистов.
- Можно ли давать архитектуру в качестве технического задания для рядовых исполнителей? Нет, в большинстве случаев это бесполезно, так как при «вхождении в IT» на разных курсах по программированию таким высоким материям не учат.
- Архитектура вашего проекта это сложный и ответственный раздел, которому нужно посвящать много времени.

Описание реализации

- Описание реализации нужно для того, чтобы у участников проекта было понимание тонкостей реализации классов, методов и структур данных.
- По собственному опыту: при интенсивной и разноплановой работе над несколькими проектами понимание собственных исходных текстов может исчезнуть за пару месяцев.
- Как нужно документировать?
  - Желательно без отрыва от исходных текстов проекта. Все благие намерения по написанию документации в отдельном и красивом документе как правило заканчиваются тем, что из-за большой нагрузки документ забрасывается и очень быстро теряет актуальность.

Виды тестирования

Раздел про тестирование

- Мы будем использовать только три вида тестов:
  - Модульное (юнит-тесты отдельных классов, механизм тестирования встроен в InteliJ IDEA)
  - Интеграционное (тестирование подсистем, микросервисов, библиотек,...)
  - Системное (все целиком)
- Для интеграционного и системного тестирования вам понадобится разработать тестовое инструментальное обеспечение на языке Kotlin/Go/Python

Список литературы

- Вы не можете что либо написать не делая ссылок на источники по следующим причинам:
  - Нет смысла повторять то, что уже написано другими. Достаточно сослаться на статью или книгу.
  - Цитируя чужие тексты без указания источника вы занимаетесь плагиатом. Например, с дипломом такое цитирование закончится тем, что вы не пройдете через систему «Антиплагиат». Аналогичная проблема может возникнуть при написании статьи или книги.
  - Читателю может понадобится дополнительная информация из первоисточника, который вы указали.
  - Читателю может понадобиться доказательство ваших утверждений.
- ГОСТ Р7.0.80—2023 Система стандартов по информации, библиотечному и издательскому делу БИБЛИОГРАФИЧЕСКАЯ ЗАПИСЬ

Примеры библиографических записей

- Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем : учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2023. — 432 с. — (Высшее образование). — ISBN 978-5-534-07604-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/513067 (дата обращения: 10.09.2023).. Общие требования и правила составления
- Исследовано в России [Электронный ресурс]: многопредмет. науч. журн. / Моск. физ.-техн. ин-т. – Электрон. журн. – Долгопрудный: МФТИ, 1998. – режим доступа к журн.: http://zhurnul.milt.rissi.ru (дата обращения: 06.05.2023)
- Lindell I. V., Sihvola A. H., Tretyakov S. A., Viitaten A. J. Electromagnetic Waves in Chiral and Bianisotropic Media. Boston — London: Artech House, 1994.

Заключение

- В заключении приводятся результаты и достижения, которые вы получили в процессе работы над проектом.
- Результат это обычно выполненная задача, указанная в начале документа. Например:
  - Собраны сухие ветки в достаточном количестве.
  - Получен огонь с помощью куска кремня и рессоры от тойоты.
  - Разведен костер.
  - Работа с костром показала, следующее:
    - Спать зимой на снегу вокруг костра неудобно, так как спереди жарко, а сзади холодно.
    - Волки боятся огня, но они громко воют и страшно смотрят светящимися глазами из темноты.
    - Запас дров нужно рассчитывать таким образом, чтобы с одной стороны не слишком утомить себя заготовкой, а с другой, чтобы хватило дров для поддержания горения до утра.

База знаний в Obsidian

- Вы записываете в базу знаний то, что вам непонятно:
  - новый для себя язык программирования, скриптовой язык или DSL,
  - библиотеку,
  - технологию,
  - фреймворк,
  - концепцию и т.п.
- Вы будете находить информацию с помощью нейросетей, в большинстве случаев вам будет достаточно бесплатного DeepSeek
- Вы будете отвечать на вопросы по своей базе знаний в конце обучения, вашу базу знаний должны изучить все члены рабочей группы. Информациями стало очень мнорго. **Как уже говорилось выше, теперь проблема состоит не в том, чтобы найти информацию, а в том, чтобы ее успеть понять.**

Требования к исходным текстам

- Основная часть исходных текстов пишется с помощью ИИ агентов
- Вы должны понимать то, что написал ИИ агент

Спасибо за внимание!