Введение:

Данный отчет представляет собой итог выполнения цикла практических работ. Основной целью проведенных работ являлось закрепление теоретических знаний и приобретение практических навыков. В ходе работы были выполнены практические в количестве 9 работ.

СОДЕРЖАНИЕ

**1.**

Тема: Инструментальные средства разработки программного обеспечения

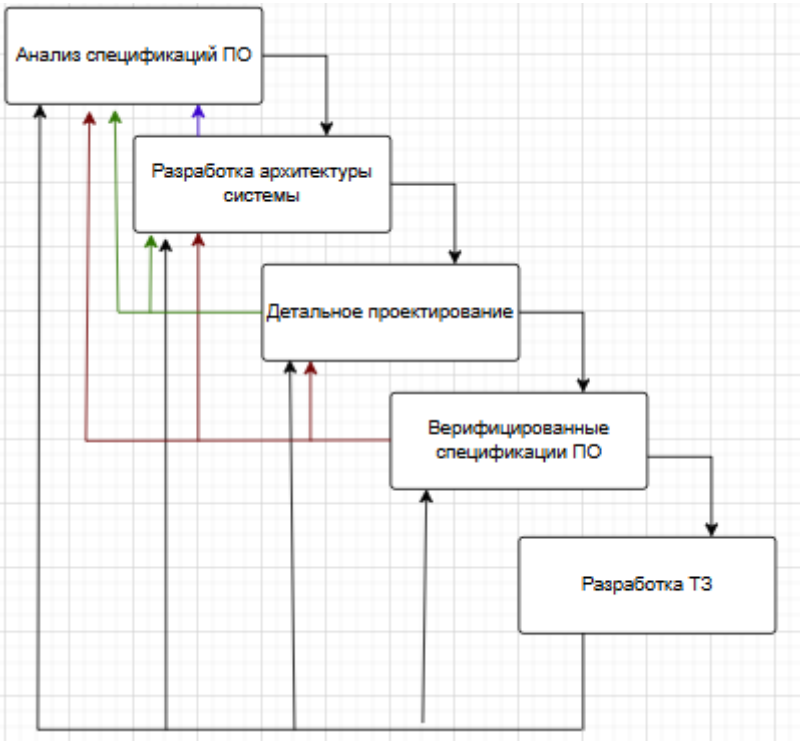
Цель работы: Изучить современные инструментальные среды разработки программного обеспечения.

Основные понятия темы

Программное обеспечение (ПО) — совокупность программ, процедур и документации, обеспечивающих функционирование вычислительной системы и выполнение определённых функций.

Инструменты разработки ПО — программное обеспечение, используемое разработчиками для написания, тестирования, отладки и управления проектами.

Схема циклической модели проектирования ПО:




Состав современных систем программирования:

1. Интегрированные среды разработки (IDE)
2. Компиляторы и интерпретаторы
3. Библиотеки и фреймворки
4. Средства управления версиями
5. Отладчики и профилировщики
6. Система автоматической сборки и деплоймента (CI/CD)

Основные функции современных компиляторов:

- Лексический анализ
- Синтаксический анализ
- Семантический анализ
- Оптимизация
- Генерация кода
- Линковка

Основные функции современных интерпретаторов:

- Лексический анализ
- Синтаксический анализ
- Выполнение
- Управление памятью

Схему процесса описания реализации программного кода с подробным описанием каждого этапа:

1. Написание программы в соответствии с требованиями и дизайном, определёнными на предыдущих этапах.
2. Создание модулей, компонентов и функциональных частей программы.
3. Документирование кода — создание документации по коду с инструкциями для других разработчиков, а также руководства по функциям приложения для конечных пользователей.

Современные средства программирования

| Средство | Краткое описание |
|---|---|
| Python | Высокоуровневый язык общего назначения с простым синтаксисом и богатой экосистемой библиотек. |
| C++ | Язык низкого уровня, используется для высокопроизводительных приложений, драйверов и встроенных систем. |
| VSS | Версионная система SourceSafe от Microsoft, позволяющая отслеживать изменения в проекте. |
| MS Visual Studio | Интегрированная среда разработки от Microsoft для .NET Framework, поддерживающая языки C#, VB.NET, F# и др. |
| Oracle | Реляционная СУБД корпоративного класса с поддержкой сложных запросов и транзакционной целостности. |
| MS SQL Server | Реляционная база данных от Microsoft, широко применяемая в корпоративных приложениях Windows. |
| MySQL | Открытая реляционная СУБД, используемая в веб-приложениях и системах баз данных среднего масштаба. |

Этапы проектирования приложений:

1. Анализ и исследование:
2. Проектирование UX/UI:
3. Техническое проектирование:
4. Детальная спецификация:
5. Планирование разработки:
6. Утверждение и передача

Краткий список нотаций для этапа проектирования:

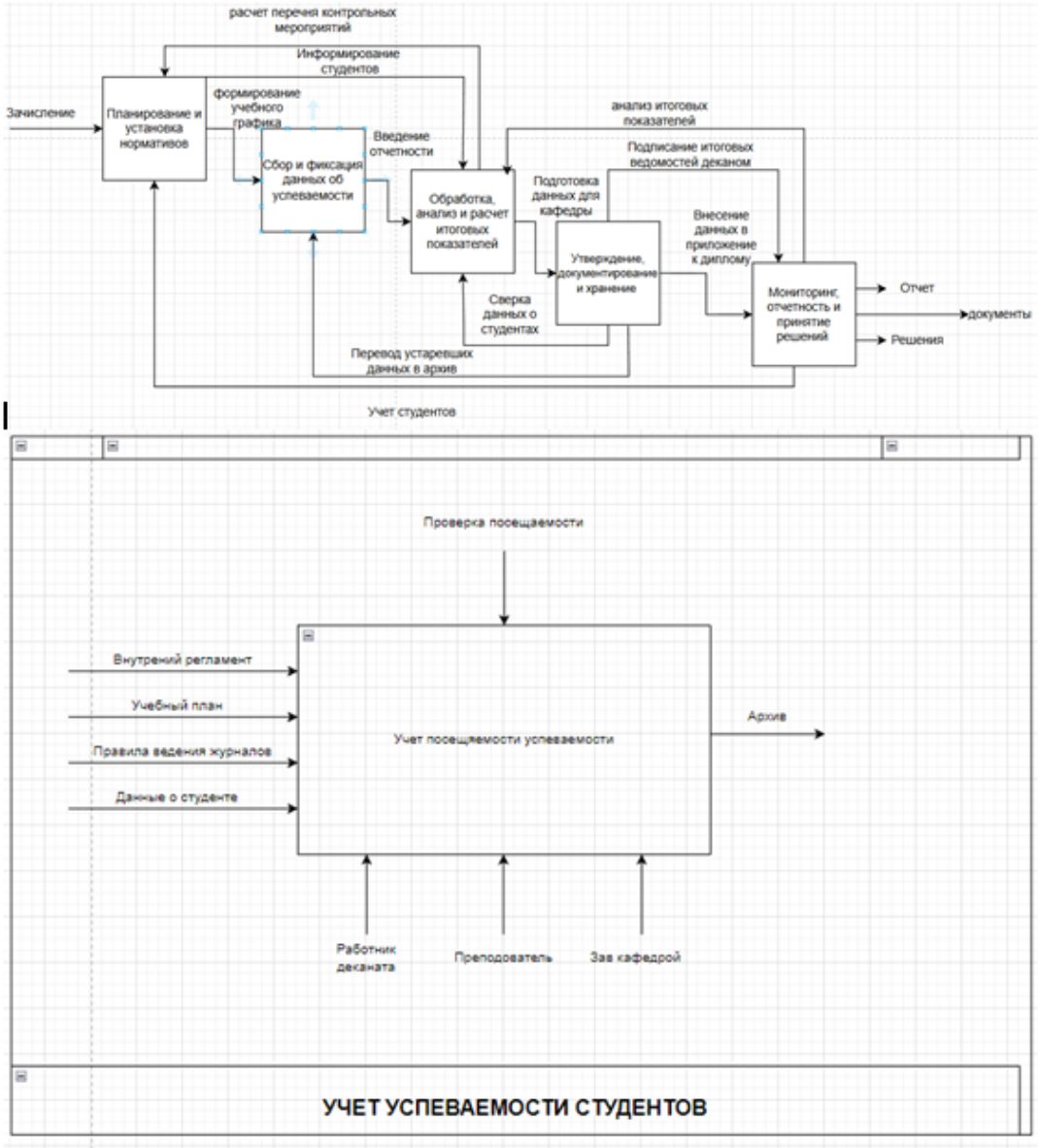
- UML.
- ER
- DFD.
- BPMN.
- SysML.

- IDEF.

Средства для этапа проектирования:

- CASE-системы
- Графические редакторы
- Редакторы диаграмм
- Трекеры требований
- Средства совместной работы
- Архитектурные конструкторы

Выводы: мы выполнили работу на тему Инструментальные средства разработки программного обеспечения.



Разработка и оформление технического задания

Техническое задание на разработку автоматизированной системы "Учет обучающихся студентов"

1. Введение

Документ регламентирует разработку программного продукта «Автоматизированная система учета успеваемости студентов», назначение которого — автоматизировать учет успеваемости, упростив управление образовательным процессом и повысив качество принимаемых решений.

2. Наименование и область применения

Название: Автоматизированная система учета успеваемости студентов (АСУС). Применима в учреждениях среднего и высшего профессионального образования, реализующих образовательные программы бакалавриата, специалитета, магистратуры и аспирантуры.

3. Основание для разработки

Основанием является приказ ректора № 65 от «12.02» 2024 г., предписывающий автоматизировать процесс учета успеваемости.

4. Назначение разработки

Комплекс предназначен для автоматизации учета успеваемости студентов, облегчения работы преподавателей и администрации вуза, повышения прозрачности отчетности и качества принятых решений.

5. Технические требования к программе

5.1 Состав функций системы

Учёт студентов и преподавателей;

Электронный личный кабинет студента;

Статистика посещаемости занятий;

Выставление промежуточных оценок;

Расчёт индивидуального индекса успеваемости;

Регулирование доступа к модулям системы;

Отправка уведомлений и сообщений;

Предоставление рекомендаций студентам;

Формирование и публикация отчетов.

5.2 Пользовательские интерфейсы

Удобный веб-интерфейс, совместимый с современными браузерами Chrome, Firefox, Safari, Edge.

5.3 Поддерживаемые операционные системы и оборудование

Совместимость с серверами под управлением ОС Linux и Windows Server.

5.4 Прочие требования

Соблюдение норм информационной безопасности и защиты персональных данных в соответствии с российским законодательством.

6. Технико-экономические показатели

Экономия трудовых затрат на документооборот;

Повышение точности расчетов успеваемости;

Улучшение управляемости образовательным процессом;

Срок окупаемости — минимум 2 года.

7. Этапы разработки

Подготовительный этап: разработка ТЗ, планирование.

Основной этап: проектирование, программирование, тестирование.

Завершающий этап: опытная эксплуатация, сдача-приемка.

8. Контроль и приемка

Испытания проводятся на каждом этапе разработки, проверяются программа внутренним аудитом, организуются приемочные испытания совместно с заказчиком.

Вывод: Мы сделали техническое задание на тему: “Учет обучающихся”.

Выполнение заданий:

1. Жизненный цикл программного продукта (ПП), критерии качества ПП, виды ПО, стадии разработки ПП

Жизненный цикл программного продукта (ПП) – последовательность этапов развития программного обеспечения, начиная с идеи до завершения поддержки. Включает этапы:

Планирование и анализ требований.

Проектирование.

Реализация (кодирование).

Тестирование.

Эксплуатация и поддержка.

Завершение жизненного цикла конец эксплуатации.

Критерии качества ПП: надежность, удобство использования, производительность, безопасность, переносимость, сопровождаемость.

Виды ПО:

Прикладное программное обеспечение.

Системное программное обеспечение.

Инструментальное программное обеспечение.

Стадии разработки ПП:

1.Постановка задачи и анализ требований.

1. Проектирование.
2. Кодирование.
3. Тестирование.
4. Документирование.
5. Интеграция и развертывание.
6. Обслуживание и сопровождение.

2. Разработка требований (определение, виды работ)

Определение: процесс выявления и документирования функций и характеристик, которым должна соответствовать система или продукт, чтобы удовлетворять потребности клиентов и бизнеса.

Виды работ:

Сбор требований.

Анализ и проверка требований.

Спецификация требований.

Утверждение требований.

3. Определения основных типов требований

Пользовательские требования: Определяют функциональные и качественные ожидания пользователей относительно будущего продукта. Обычно задаются на высоком уровне абстракции.

Системные требования: Подробные спецификации, определяющие конкретные функции и ограничения системы, необходимые для выполнения указанных пользовательских требований.

Проектная системная спецификация: Документация, содержащая детальные технические спецификации системы, включая структуру компонентов, интерфейсы, используемые технологии и стандарты.

4. Определения ключевых видов требований

Функциональные требования: Четко обозначают, какие операции должна поддерживать система.

Нефункциональные требования: устанавливают дополнительные условия функционирования, такие как производительность, надежность, удобства использования и другие метрики качества.

5. Таблица сравнения моделей разработки

| № п/п | Модель разработки и разработчики | Особенности | + | - |
|---|---|---|---|---|
| 1 | SADT-Дугласс Росс | Функциональные диаграммы | Четкость | Жесткость |
| 2 | CASE-Джеймс Мартин | Автоматизация проектирования | Скорость | Дорого |
| 3 | OOAD-Грэди Буч | Работа с объектами | Гибкость | Сложность |
| 4 | UML-“Три амиго” | Визуальное моделирование | Наглядность | Избыточность |

Выбор модели: я выбрал бы CASE - модель, поскольку она обеспечивает большую скорость, что особенно полезно в современных условиях неопределенности.

6. Группа разработчиков

Группа разработчиков состоит из ролей:

Руководитель проекта.

Архитектор.

Разработчик.

Тестировщик.

Специалист по качеству.

Менеджер по требованиям.

Заключение

Данная практика позволила углубленно изучить методики выработки требований к программному обеспечению, рассмотреть основные концепции проектирования и выбора подходящих моделей разработки. Применение полученных знаний обеспечит эффективное создание качественного программного продукта.

1. Проектирование ПО

процесс разработки структуры и внутренних связей компонентов системы для достижения поставленных целей и удовлетворения потребностей пользователей.

2. Концептуальная архитектура

общая концепция устройства системы, определяющая ключевые элементы и направления дальнейших разработок.

3. Архитектурный стиль

набор подходов и принципов организации компонентов системы (например, MVC, микросервисы, SOA).

4. Пилотная архитектура / Базовая архитектура

начальная версия архитектуры, создающая основу для последующего расширения и доработки системы.

5. Модуль

отдельно реализуемый элемент программы, решающий конкретную задачу и взаимодействующий с другими модулями через чётко определённые интерфейсы.

6. Компонент

логически завершённая часть программы, обладающая собственной функциональностью и возможностью повторного использования.

7. Фреймворк

готовый комплект инструментов и конструкций, позволяющий ускорить разработку программного обеспечения.

8. Слабая связанность

минимальное количество зависимостей между отдельными частями системы, способствующее легкости изменений и развитию.

9. Сквозная функциональность

общие аспекты системы, влияющие сразу на несколько её частей (например, безопасность, журналирование, мониторинг).

10. Портирование ПО

адаптация программы для работы на другой платформе или среде исполнения.

11. Программный

код последовательность команд на языке программирования, исполняемая компьютером для выполнения конкретных задач.

12. Структура кода

организация и размещение элементов программного кода, облегчающие чтение, поддержку и развитие программы.

Задание 1. Термины тестирования и разработки ПО

1. Комплексное тестирование

Процедура проверки поведения готовой системы на соответствие спецификациям и требованиям, проводимых после завершения интеграционного тестирования. Цель комплексного тестирования — убедиться, что программа функционирует должным образом в реальной среде.

2. Отладка

Процесс нахождения и исправления ошибок в программе. Отладка подразумевает пошаговое исследование и изучение состояния программы для точного выявления причины возникшей неисправности.

3. Тест

Контролируемое действие, осуществляемое для проверки правильности работы программы или компонента системы. Может проводиться как вручную, так и автоматически.

4. Верификация

Процесс подтверждения того, что продукт соответствует заранее определенным требованиям и спецификациям. Верификация включает проверку документации, процедур и артефактов разработки.

5. Валидация

Проверка того, насколько разработанный продукт действительно полезен и решает задачи пользователей. То есть оценивается пригодность и полезность продукта, а не соответствие формальным требованиям.

6. Этапы процесса тестирования

Процесс тестирования проходит последовательно через фазы:

Планирование и подготовка.

Проектирование тестов.

Выполнение тестов.

Анализ результатов и коррекция.

Репортаж и документирование ошибок.

7. Цикл тестирования

Стандартный цикл, включающий многократное выполнение этапов тестирования (от первоначального тестирования до повторного тестирования после исправления ошибок). Каждая итерация направлена на повышение качества продукта.

8. Модульное тестирование

Метод тестирования отдельных модулей или компонентов программы, чтобы удостовериться, что каждая отдельная часть работает правильно. Применяется на ранних этапах разработки.

9. Интеграционное тестирование

Тестирование, которое проверяет, насколько успешно интегрируются два или более модуля программы. Проводится после успешного модульного тестирования.

10. Системное тестирование

Вид тестирования, при котором проверяется полная система или продукт, работающий в производственной среде, на соответствие требованиям и спецификациям.

11. Выходное тестирование

Последний этап тестирования перед выпуском продукта. Цель — подтвердить, что продукт стабилен и готов к выходу на рынок.

12. Программная ошибка (bug)

Неправильное поведение программы вследствие неверного кода, логических ошибок или неправильного использования ресурсов.

13. Регрессионное тестирование

Тип тестирования, который проводится после внесения изменений в программу, чтобы проверить, что старые функции продолжают работать корректно и не появились новые ошибки.

14. Тестирование «черного ящика» (Black Box)

Метод тестирования, при котором внутренняя структура программы неизвестна, а тесты строятся на основе внешнего поведения и ожидаемых реакций системы на определенный ввод.

15. Тестирование «белого ящика» (White Box)

Способ тестирования, при котором известна внутренняя структура программы, и тесты создаются с учётом её устройства и кода.

16. Трассировка

Процесс наблюдения за выполнением программы с фиксацией состояний, промежуточных результатов и путей выполнения. Используется для диагностики ошибок и изучения логики программы.

17. Тестовые сценарии

Формализованные шаги и процедуры, следуя которым тестировщик проверяет правильность работы программы.

Задание 2. Три закона программотехники

Закон неисправимого совершенствования: любая сложная программа имеет недостатки и уязвимости, устранить абсолютно все ошибки практически невозможно.

Закон бесполезности комментариев: лучше писать понятный и самодокументированный код, чем полагаться на чрезмерное комментирование.

Закон обратной связи: программируя, обязательно оставляйте отзыв самому себе в будущем («пиши комментарий, иначе завтра забудешь, зачем это написал»).

Задание 3. Программа на Python

Напишем программу расчета площади треугольника по трем сторонам. Далее реализуем набор тестов и контроль вводимых данных.

Программа на Python:

```python
import math

def calculate_triangle_area(a, b, c):
    if a <= 0 or b <= 0 or c <= 0:
        return "Некорректные данные!"
    elif a >= b+c or b >= a+c or c >= a+b:
        return "Невалидный треугольник!"
    else:
        s = (a + b + c) / 2
        area = math.sqrt(s*(s-a)*(s-b)*(s-c))
        return f"Площадь треугольника равна {area:.2f}"

# Ввод данных
try:
    side_a = float(input("Введите длину стороны A: "))
    side_b = float(input("Введите длину стороны B: "))
    side_c = float(input("Введите длину стороны C: "))
except ValueError:
    print("Ошибка! Нужно ввести числа.")
else:
    result = calculate_triangle_area(side_a, side_b, side_c)
    print(result)
```

Тестовые данные:

| Значение A | Значение B | Значение C | Ожидаемый результат | Фактический результат |
|---|---|---|---|---|
| 3 | 4 | 5 | Площадь треугольника = 6.00 | Площадь треугольника = 6.00 |
| 1 | 1 | 1 | Площадь треугольника = 0.43 | Площадь треугольника = 0.43 |
| 0 | 3 | 4 | Некорректные данные! | Некорректные данные! |
| 10 | 1 | 1 | Невалидный треугольник! | Невалидный треугольник! |
| abc | def | ghi | Ошибка! Нужно ввести числа. | Ошибка! Нужно ввести числа. |

Рекомендации по исправлению ошибок:

1. Ввести дополнительную проверку ввода для исключения букв и символов.
2. Добавить плавающую запятую для ввода десятичных дробей.
3. Дополнительно проверять равенство суммы двух сторон третьей стороне для лучшего контроля ошибок.

Задание 4. Сопровождение ПО (ИС)

Сопровождение ПО

Продолжающаяся активность после поставки программного обеспечения, включающая:

Консультативную помощь пользователям.

Обновление программных компонентов.

Исправление ошибок и внесение улучшений.

Обучение пользователей новым возможностям.

Варианты сопровождения:

Корректирующее сопровождение: исправление выявленных ошибок и недочетов.

Административное сопровождение: мониторинг инфраструктуры и настроек системы.

Адаптирующее сопровождение: изменение конфигурации ПО под изменившиеся требования.

Расширяющее сопровождение: добавление новых функций и возможностей.

1.1. Организационная структура — внутреннее устройство и распределение полномочий в организации, характеризующее взаимосвязи между отдельными звеньями и людьми, участвующими в управлении деятельностью фирмы или учреждения.

1.2. Структура управления — конфигурация подразделений и должностных позиций, составляющих основу организации, определяющая вертикальные и горизонтальные связи подчинённости и власти.

1.3. Элемент организационной структуры — это подразделение, отдел, служба или сотрудник, выполняющий конкретные обязанности и функции в рамках установленной структуры организации.

1.4. Уровни (ступени управления) — иерархические ступени управления организацией, распределённые по степени влияния на принимаемые решения: высшее руководство, среднее звено менеджеров и исполнители.

1.5. Регламентирование — установление правил и порядка выполнения работ, закреплённое официальными актами, процедурами и положениями, обязательными для соблюдения всеми участниками организации.

1.6. Нормирование — установка норм, показателей и ограничений, регулирующих объём и качество выполнения производственных или иных задач в целях повышения эффективности и управляемости организации.

1.7. Инструктирование — передача информации и разъяснений сотрудникам по вопросам выполнения поручений, действий в чрезвычайных ситуациях, регламента выполнения задач и обращения с оборудованием.

1.8. Делегирование — передача руководителем части своих полномочий и ответственности, подчинённым для самостоятельного выполнения задач и принятия решений в установленных границах.

1.9. Полномочия — права и полномочия, официально установленные лицу или должности, предоставляющие возможность действовать и принимать решения в интересах организации.

1.10. Ответственность — обязанность отвечать за последствия принятых решений, выполненных действий или невыполнения возложенных обязательств, предусмотренная соответствующими

1. Справочное руководство на программный продукт Wildberries

Справочное руководство (также называемое Reference Manual) предоставляет точную и подробную информацию о каждом элементе программного продукта, описании его методов, функций, параметров и возвращаемых значениях. Данное руководство нацелено на профессиональных пользователей и разработчиков, которые хотят глубоко разбираться в особенностях и деталях работы сервиса Wildberries.

Ключевые разделы:

Описание API-интерфейсов для работы с товарами, категориями, ценами, статусом заказов и логистическими услугами.

Параметры запросов и их влияние на возвращаемый результат.

Ошибки и коды возврата HTTP-запросов.

Доступные методы и их аргументы, включая правила передачи параметров.

Типы данных и формы сериализации данных.

Лучшие практики и советы по повышению производительности запросов.

Формы подачи материалов:

Табличная форма для описания методов и параметров.

Примерные запросы и ответы.

Иллюстративные фрагменты кода для демонстрации использования API.

2. Руководство пользователя Wildberries

Руководство пользователя (User Guide) предназначено для обычных покупателей и продавцов на маркетплейсе Wildberries. Оно помогает ориентироваться в сервисе, знакомит с ключевыми возможностями и даёт пошаговые инструкции по выполнению задач.

Ключевые разделы:

Начало работы: регистрация аккаунта продавца или покупателя.

Настройки профиля и кабинета продавца: создание карточек товаров, работа с каталогом, ценообразование.

Управление продажами: мониторинг заказов, обработка возвратов, расчет прибыли.

Покупателям: инструкция по выбору товара, оформлению заказа, применению скидок и возврату товаров.

Оплата и доставка: доступные способы оплаты, расчёт стоимости доставки, политика возврата денег.

Проблемы и решение: частые вопросы и инструкции по устранению трудностей.

Формы подачи материалов:

Текстовые инструкции с иллюстрациями и скриншотами.

Пошаговые руководства с пояснением простых действий.

Видеоинструкции и интерактивные демоверсии.

3. Руководство программиста Wildberries

Руководство программиста (Developer’s Guide) рассчитано на разработчиков и специалистов, желающих интегрироваться с сервисом Wildberries или расширить его функциональность. Оно охватывает глубокие аспекты интеграции, обновления и кластеризации API и SDK.

Ключевые разделы:

Архитектура API Wildberries: принцип работы, использование запросов и обработка ответов.

Библиотеки и SDK для интеграции с системой.

Паттерны и антипаттерны при взаимодействии с API.

Шаблон подключения и примеры интеграции.

Принципы обработки ошибок и исключений.

Примеры типичной интеграции с магазинами и CRM-системами.

Методология использования сторонних сервисов и расширений.

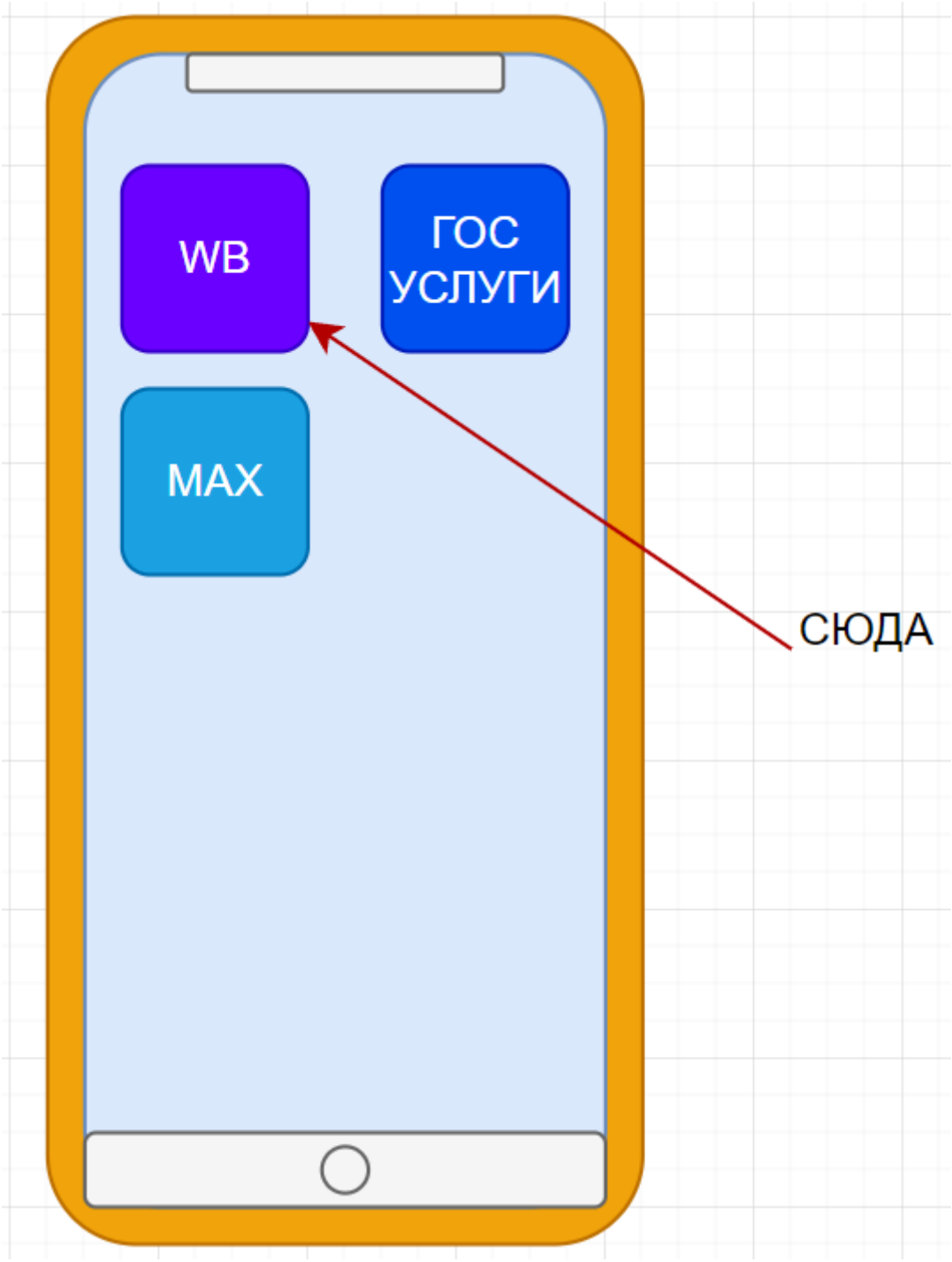
Формы подачи материалов:

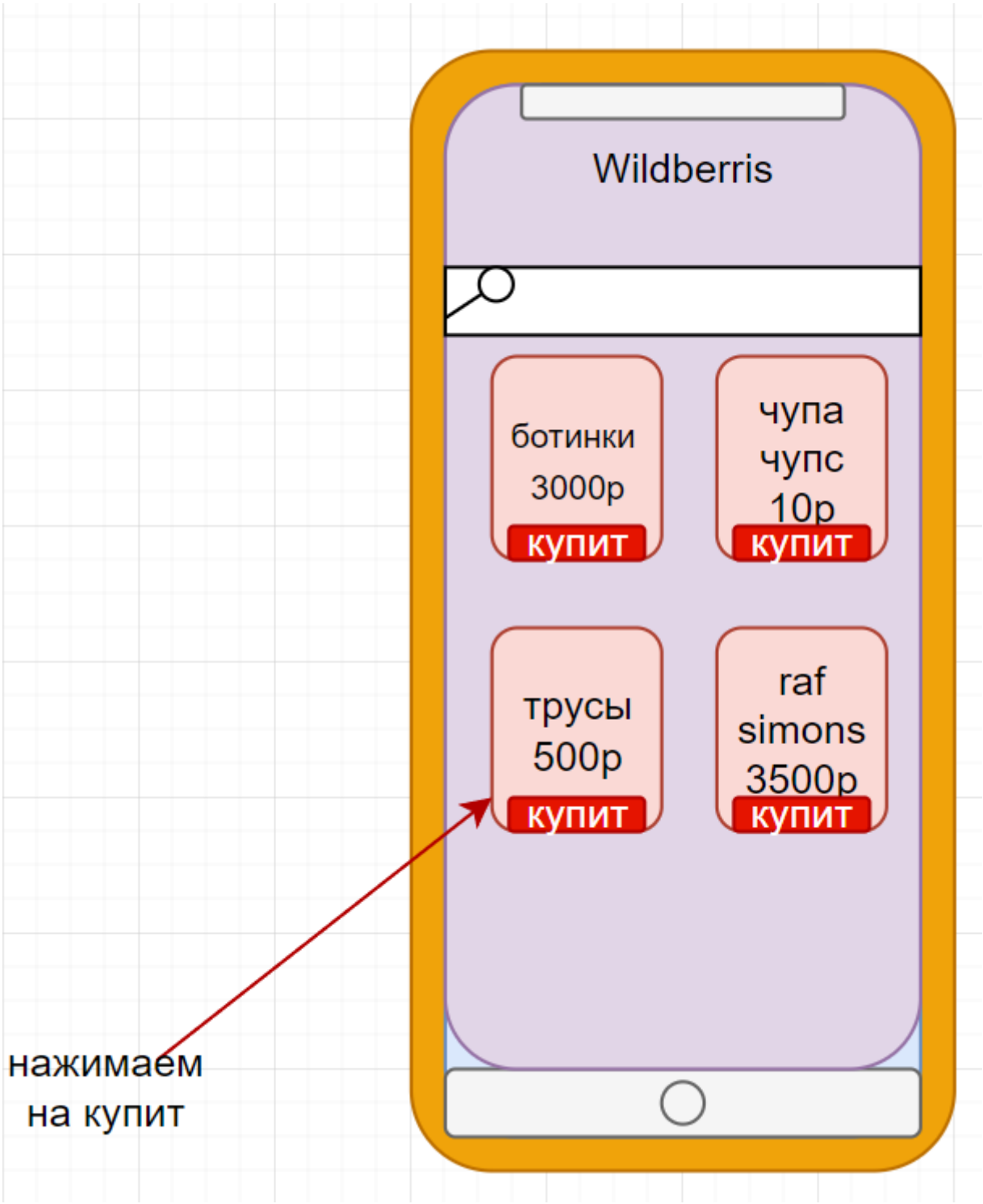
Глубокая теория и объяснения по различным частям API.

Примеры кода на популярных языках программирования (Python, JavaScript, PHP и др.).

Демонстрация способов обработки сложных ситуаций

Руководство пользователя для любого выбранного Маркетплейса


WB
ГОС УСЛУГИ
MAX
СЮДА




**9.**

По терминам:

- Сертификация в IT – подтверждение, что процессы или продукты соответствуют стандартам (например, ГОСТам).
- Лицензия на ПО – разрешение на использование программы на определённых условиях, а не право собственности на неё.

- Права:
- Исключительные: правообладатель не может использовать ПО сам и выдавать лицензии другим.
- Неисключительные (простые): правообладатель сохраняет все права и может выдавать лицензии кому угодно (самый частый вариант).
- Лицензии для кода:
- GNU GPL: свободное использование, но если модифицируете код, ваш продукт тоже должен быть с открытым кодом.
- FreeBSD: свободное использование, можно встраивать в закрытые коммерческие продукты без открытия своего кода.
- Типы ПО:
- Бесплатное: бесплатно, но код закрыт (например, Adobe Reader).
- Условно-бесплатное: бесплатный пробный период или версия с ограничениями (WinRAR).
- Коммерческое: платное, код закрыт (Microsoft Windows).
- OEM/BOX: OEM – дешевле, вшито в устройство; BOX – дороже, «коробочная» самостоятельная версия.
- Ответственность в РФ: регулируется Гражданским кодексом и КоАП. За использование нелицензионного ПО грозят крупные штрафы (ст. 7.12 КоАП РФ), а в крупных размерах – уголовная ответственность (ст. 146 УК РФ).

2. Методы оценки затрат (примеры):

- Основные затраты: зарплата команды (основная статья), оборудование, софт, маркетинг.
- Метод аналогов: ориентируемся на прошлые похожие проекты.
- Расчёт: «Похожий проект сделали за 1000 часов. Новый сложнее на 20%, значит, ~1200 часов».
- Особенность: быстро, но неточно.

- Метод функциональных точек: считаем не часы, а «баллы» за функции программы.
- Расчёт: проекту начислили 100 точек. 1 точка = 20 часов работы. Итого: 100 * 20 = 2000 часов.
- Особенность: объективнее, но сложнее в подсчёте.
- Метод декомпозиции: дробим большой проект на мелкие задачи и оцениваем каждую.
- Расчёт: модуль А – 40 ч., модуль Б – 80 ч., модуль В – 120 ч. Итого: 240 ч.
- Особенность: Точно, но требует детального ТЗ и много времени.

3. Пояснительная записка (согласно ГОСТ):

Разработана в соответствии со структурой, аналогичной ГОСТ 7.32-2017. Основные разделы и их содержание:

1) Введение: актуальность, цель и задачи проекта.

2) Основная часть:

- Анализ и ТЗ: Обзор аналогов, обоснование разработки, требования.
- Проектирование: Выбор технологий, проектирование архитектуры, базы данных, интерфейсов.
- Реализация: Описание готовых модулей, листинги ключевого кода.

3) Заключение: итоги работы, выводы, достигнута ли цель.

Результат работы:

- Программный продукт собран и протестирован.
- Для защиты подготовлена краткая презентация, отражающая основные этапы работы: от постановки задачи до демонстрации работающего приложения.
- Продукт готов к защите.