**НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ «СИНЕРГИЯ»**

|  |  |  |
| --- | --- | --- |
| **Факультет/Институт** |  | Информационных технологий |
|  |  | (наименование факультета/ Института) |
| **Направление/специальность** |  | Информационные системы и технологии |
| **подготовки:** |  | (код и наименование направления /специальности подготовки) |
| **Форма обучения:** |  | очная |
|  |  | (очная, очно-заочная, заочная) |
|  |  |  |

**Лабораторный практикум № 7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **по дисциплине** |  | Программирование на IOS | | |
|  | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | **Петухов Никита Максимович** |  |  |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | **группа ДБИ-301рки** |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  | Сибирев Иван Валерьевич |  |  |
|  |  | (ФИО) |  | (подпись) |

**Москва 2024 г.**

Успешная разработка приложений зависит от грамотного выбора архитектурных паттернов. Ключевые подходы, обеспечивающие поддерживаемый и масштабируемый код, включают SOLID, SOA, MVC и MVVM. **SOLID:** Этот набор принципов объектно-ориентированного программирования (ООП) фокусируется на создании гибких систем. В него входят: Принцип единственной ответственности (каждый класс имеет одну задачу), Принцип открытости/закрытости (расширяемость без изменения исходного кода), Принцип подстановки Лисков (замена подклассов без нарушения работы), Принцип разделения интерфейсов (использование только необходимых интерфейсов) и Принцип инверсии зависимостей (зависимость от абстракций, а не конкретных реализаций). **Сервис-ориентированная архитектура (SOA):** SOA структурирует приложение как набор независимых сервисов, взаимодействующих через интерфейсы. Это упрощает интеграцию, позволяет независимо обновлять и масштабировать сервисы, и повышает гибкость кода. **MVC (Model-View-Controller) и MVVM (Model-View-ViewModel):** Это паттерны, разделяющие приложение на компоненты: MVC разделяет на модель (данные и логика), представление (отображение данных) и контроллер (обработка ввода). MVVM добавляет ViewModel – посредник между моделью и представлением, упрощая тестирование и поддержку благодаря четкому разделению обязанностей и использованию двустороннего связывания данных. Несоблюдение принципов SOLID в MVC может привести к избыточно сложным контроллерам.

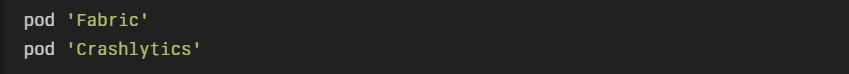
Протоколы в Swift позволяют определять интерфейсы для классов без необходимости указывать конкретную реализацию. Шаблон делегат используется для передачи информации между объектами:

objectivec  
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Fabric и Crashlytics предоставляют инструменты для мониторинга производительности приложения и отслеживания сбоев. Для интеграции в проект выполните следующие шаги:

1. **Установка CocoaPods**:  
   Добавьте в ваш Podfile:

ruby  


1. **Инициализация в AppDelegate**:  
   В AppDelegate.m добавьте код для настройки Fabric:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. **Отслеживание сбоев**:  
   Crashlytics автоматически отслеживает сбои в приложении. Вы также можете вручную регистрировать события:

Objectivec



Вывод: Создание архитектуры приложения требует понимания различных паттернов проектирования и их применения в зависимости от требований проекта. Использование принципов SOLID, SOA, MVC и MVVM помогает создавать гибкие и поддерживаемые приложения. Интеграция протоколов и шаблона делегат улучшает взаимодействие между компонентами, а использование Fabric и Crashlytics обеспечивает мониторинг производительности и стабильности приложения.