

Министерство образования и науки РФ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Омский государственный технический университет»

Кафедра «Автоматизированные системы обработки информации и
управления»

Отчёт по дисциплине
«Современные инструментальные средства разработки программного
обеспечения»

Выполнил:

Студент гр. ПИН-201

Голуб А.Е.

(подп., дата)

Проверил:

Старший преподаватель

Кабанов А.А.

(подп., дата)

Омск 2024

Задание

Получить опыт работы с большой языковой моделью Chat-GPT и нейросетью-художником Midjourney. Используя ее инструментарий сгенерировать полезные данные для выпускной квалификационной работы.

Ход выполнения лабораторной работы

План работы над проектом "Автоматизированная информационная система запросов и планирования обслуживания клиентов гостиницы".



ChatGPT

Разработка "Автоматизированной информационной системы запросов и планирования обслуживания клиентов гостиницы" - это серьезный и многозадачный проект. Вот предложенный план работы, который может помочь вам структурировать процесс:

1. **Определение требований:**
 - Взаимодействие с клиентами и персоналом гостиницы.
 - Определение основных функциональных требований к системе.
2. **Анализ существующих систем:**
 - Исследование существующих решений на рынке.
 - Определение их преимуществ и недостатков.
3. **Создание технического задания:**
 - Формирование подробного технического задания с учетом выявленных требований.
 - Определение технологий, используемых для разработки.
4. **Проектирование системы:**
 - Разработка архитектуры системы.
 - Создание диаграммы базы данных.
 - Проектирование интерфейса пользователя.
5. **Разработка:**
 - Создание основных модулей системы.
 - Разработка базы данных и ее интеграция с системой.
 - Тестирование каждого модуля отдельно.
6. **Интеграция и тестирование системы:**
 - Интеграция всех модулей в единую систему.
 - Проведение системных тестов для проверки работы системы в целом.
7. **Тестирование безопасности:**
 - Проверка системы на уязвимости.
 - Разработка механизмов безопасности.
8. **Обучение персонала:**
 - Подготовка обучающих материалов.
 - Проведение обучения персонала по использованию новой системы.
9. **Внедрение и поддержка:**
 - Последовательное внедрение системы в рабочее окружение.
 - Проведение мониторинга и поддержка после внедрения.
10. **Оценка эффективности:**
 - Сбор обратной связи от пользователей.
 - Анализ эффективности системы и, при необходимости, внесение корректив.
11. **Оптимизация и дальнейшее развитие:**
 - Идентификация областей для оптимизации.
 - Планирование дальнейшего развития системы с учетом изменяющихся потребностей.
12. **Документирование:**
 - Создание полной документации по системе для последующего использования и поддержки.

Убедитесь, что весь процесс разработки поддерживается регулярными совещаниями, отчетами о прогрессе и взаимодействием с заказчиком для выяснения дополнительных требований и изменений в процессе разработки.



Основные функциональные требования "Автоматизированная информационная система запросов и планирования обслуживания клиентов гостиницы".



ChatGPT

Основные функциональные требования для "Автоматизированной информационной системы запросов и планирования обслуживания клиентов гостиницы" могут включать следующие:

1. **Регистрация и авторизация:**
 - Возможность регистрации новых пользователей (клиентов, персонала).
 - Авторизация для безопасного доступа к системе.
2. **Управление клиентской информацией:**
 - Хранение данных о клиентах (контактная информация, предпочтения, история заказов).
 - Возможность быстрого поиска и обновления информации.
3. **Онлайн-бронирование:**
 - Механизм онлайн-бронирования номеров для клиентов.
 - Отображение актуальной доступности и цен.
4. **Управление номерами и услугами:**
 - Возможность добавления, редактирования и удаления информации о номерах и предоставляемых услугах.
 - Планирование занятости номеров и услуг.
5. **Система запросов:**
 - Возможность клиентов отправлять запросы на дополнительные услуги или изменения бронирования.
 - Уведомления персонала о новых запросах.
6. **Управление задачами персонала:**
 - Возможность для персонала просматривать текущие задачи и запросы клиентов.
 - Распределение задач и планирование работы.
7. **Уведомления и оповещения:**
 - Система уведомлений для клиентов и персонала о подтверждении бронирования, изменениях в заказе и других важных событиях.
8. **Система отчетности:**
 - Генерация отчетов о загрузке номеров, общем доходе, популярности услуг и других ключевых метриках.
 - Возможность настройки отчетов в соответствии с потребностями.
9. **Интеграция с финансовой системой:**
 - Возможность автоматического выставления счетов и обработки платежей.
 - Хранение и отслеживание финансовых транзакций.
10. **Безопасность и доступ:**
 - Защита конфиденциальной информации клиентов.
 - Разграничение прав доступа для различных ролей пользователей.
11. **Мобильная поддержка:**
 - Возможность доступа к системе через мобильные устройства для удобства персонала в движении.
12. **Аналитика и оптимизация:**
 - Средства аналитики для выявления тенденций, предпочтений клиентов и оптимизации бизнес-процессов.

Эти функциональные требования охватывают ключевые аспекты работы гостиницы, обеспечивая эффективное управление заказами, обслуживанием клиентов и внутренними процессами.

Необходимый стек технологий для "Автоматизированной информационной системы запросов и планирования обслуживания клиентов гостиницы".



ChatGPT

Выбор стека технологий для "Автоматизированной информационной системы запросов и планирования обслуживания клиентов гостиницы" зависит от множества факторов, включая требования проекта, бюджет, опыт команды разработки и другие аспекты. Однако, вот примерный набор технологий, который может быть использован:

1. Фронтенд (Frontend):

- **React.js** или **Angular**: Популярные библиотеки/фреймворки для построения динамических пользовательских интерфейсов.
- **HTML/CSS/JavaScript**: Основные технологии веб-разработки.

2. Бэкенд (Backend):

- **Node.js** (с использованием **Express.js**): Для создания серверной части приложения.
- **Python** (с использованием **Django** или **Flask**) или **Java** (**Spring**): Альтернативные варианты для бэкенд-разработки.

3. База данных:

- **MySQL, PostgreSQL** или **MongoDB**: Выбор зависит от структуры данных и требований к хранению информации.

4. Язык запросов к базе данных:

- **SQL** (для реляционных баз данных): Язык запросов к базам данных, таким как MySQL или PostgreSQL.
- **Mongoose** (для **MongoDB**): ORM для работы с MongoDB.

5. Архитектура приложения:

- **RESTful API**: Для взаимодействия между фронтендом и бэкендом.
- **GraphQL**: Вариант для более гибкого запроса данных.

6. Облачные сервисы:

- **AWS, Azure** или **Google Cloud**: Для хостинга, управления ресурсами и других облачных услуг.
- **Firebase**: Удобный инструмент для создания веб-приложений с бэкендом в облаке.

7. Управление состоянием:

- **Redux** (с **React**): Для эффективного управления состоянием приложения.
- **Vuex** (с **Vue.js**): Альтернатива для управления состоянием в приложениях на Vue.js.

8. Инструменты сборки и управления зависимостями:

- **Webpack, Babel**: Для сборки и транспиляции кода.
- **npm** или **Yarn**: Для управления зависимостями.

9. Тестирование:

- **Jest** (с **React**) или **Mocha/Chai** (с **Node.js**): Для тестирования кода.
- **Cypress** или **Selenium**: Для тестирования пользовательского интерфейса.

10. Система контроля версий:

- **Git**: Для управления версиями и совместной работы.

11. Инструменты развертывания (Deployment):

- **Docker**: Для контейнеризации приложения.
- **Heroku, Netlify** или **Vercel**: Простые в использовании платформы для развертывания и хостинга.

Это всего лишь предложение, и конкретные технологии могут быть адаптированы в соответствии с требованиями вашего проекта и предпочтениями команды разработки.

Модуль безопасности для "Автоматизированной информационной системы запросов и планирования обслуживания клиентов гостиницы".



ChatGPT

Модуль безопасности в "Автоматизированной информационной системе запросов и планирования обслуживания клиентов гостиницы" играет ключевую роль в защите конфиденциальности данных, обеспечении целостности системы и предотвращении несанкционированного доступа. Вот основные компоненты и функции, которые может включать в себя такой модуль:

1. **Аутентификация и Авторизация:**
 - **Аутентификация:** Проверка подлинности пользователей перед предоставлением доступа к системе. Это может включать в себя использование паролей, двухфакторной аутентификации и других методов.
 - **Авторизация:** Определение прав доступа пользователя к определенным ресурсам и функциям в системе.
2. **Шифрование данных:**
 - Шифрование чувствительных данных в базе данных, таких как личная информация клиентов, бронирования и финансовые данные. Реализация транспортного уровня шифрования (например, HTTPS) также важна.
3. **Управление сессиями:**
 - Безопасное управление сессиями пользователей, включая защиту от атак, таких как сессионное перехватывание (session hijacking).
4. **Защита от инъекций:**
 - Предотвращение атак на инъекции, такие как SQL-инъекции и инъекции кода, которые могут использоваться для взлома системы.
5. **Логирование и мониторинг:**
 - Регистрация действий пользователей и событий системы для последующего анализа и обнаружения аномалий.
 - Оповещения или автоматические реакции на подозрительную активность.
6. **Обработка и защита от ошибок:**
 - Грамотная обработка ошибок для предотвращения утечек конфиденциальной информации и других уязвимостей.
 - Защита от атак, таких как атаки на отказ в обслуживании (DoS).
7. **Управление доступом:**
 - Система управления ролями и правами доступа, которая позволяет точно настроить, какие пользователи имеют доступ к определенным функциям и данным.
8. **Обновление безопасности:**
 - Регулярные обновления системы и всех зависимостей для исправления известных уязвимостей.
9. **Тестирование на безопасность:**
 - Регулярные тесты на безопасность, включая тестирование на проникновение (pentesting) и аудит безопасности кода.
10. **Соблюдение стандартов безопасности:**
 - Соблюдение отраслевых стандартов и нормативов безопасности, таких как PCI DSS для финансовых данных.
11. **Блокировка и мониторинг неудачных попыток входа:**
 - Предотвращение несанкционированного доступа путем блокировки аккаунтов или введения временной задержки после неудачных попыток входа.
12. **Сегрегация данных:**
 - Разделение данных и функций системы с использованием принципа "минимизации привилегий".

Основные варианты использования для "Автоматизированной информационной системы запросов и планирования обслуживания клиентов гостиницы".



ChatGPT

К сожалению, в текстовом формате создать полноценную USE-CASE диаграмму сложно. Однако, я могу предоставить вам пример текстового описания основных вариантов использования (USE-CASE) для "Автоматизированной информационной системы запросов и планирования обслуживания клиентов гостиницы". Это может послужить основой для дальнейшего построения диаграммы.

1. Регистрация нового клиента:

- **Актер:** Гость
- **Описание:** Гость может зарегистрировать новый аккаунт в системе, предоставив необходимые данные.

2. Бронирование номера:

- **Актер:** Клиент
- **Описание:** Клиент может просмотреть доступные номера, выбрать подходящий и забронировать его на определенные даты.

3. Подача запроса на дополнительные услуги:

- **Актер:** Клиент
- **Описание:** Клиент может подать запрос на дополнительные услуги, такие как трансфер из аэропорта, заказ еды в номер и т. д.

4. Обработка запроса службой обслуживания:

- **Актер:** Персонал гостиницы
- **Описание:** Персонал гостиницы получает запросы от клиентов и обрабатывает их, назначая необходимые задачи.

5. Управление бронированием:

- **Актер:** Администратор гостиницы
- **Описание:** Администратор может просматривать и управлять всеми бронированиями, включая подтверждение, отмену и изменение.

6. Оплата бронирования:

- **Актер:** Клиент
- **Описание:** Клиент осуществляет оплату за бронирование через систему.

7. Генерация отчетов:

- **Актер:** Администратор гостиницы
- **Описание:** Администратор может генерировать различные отчеты о состоянии бронирований, финансовых данных и эффективности обслуживания.

8. Управление доступом:

- **Актер:** Администратор гостиницы
- **Описание:** Администратор управляет ролями и правами доступа пользователей к различным функциональным возможностям системы.

Эти USE-CASE описания могут быть дополнены и детализированы в зависимости от конкретных требований вашего проекта. После этого вы можете использовать эти описания для построения USE-CASE диаграммы с использованием UML (Unified Modeling Language).

ER-диаграмма для "Автоматизированной информационной системы запросов и планирования обслуживания клиентов гостиницы".




ChatGPT


Приведенный ниже SQL-код представляет собой простую ER-диаграмму для "Автоматизированной информационной системы запросов и планирования обслуживания клиентов гостиницы". Обратите внимание, что это упрощенная структура, и вам может потребоваться дополнить ее в соответствии с конкретными требованиями вашего проекта.


sql


Copy code


```
CREATE TABLE Clients (  
    ClientID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    ContactInfo VARCHAR(255)  
    -- Другие атрибуты клиента  
);  
  
CREATE TABLE Rooms (  
    RoomID INT PRIMARY KEY,  
    Type VARCHAR(255),  
    Cost DECIMAL(10, 2)  
    -- Другие атрибуты номера  
);  
  
CREATE TABLE Reservations (  
    ReservationID INT PRIMARY KEY,  
    CheckInDate DATE,  
    CheckOutDate DATE,  
    ClientID INT,  
    RoomID INT,  
    FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),  
    FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID)  
    -- Другие атрибуты бронирования  
);  
  
CREATE TABLE ServiceRequests (  
    RequestID INT PRIMARY KEY,  
    Description TEXT,  
    ClientID INT,  
    FOREIGN KEY (ClientID) REFERENCES Clients(ClientID)  
    -- Другие атрибуты запроса на услугу  
);  
  
CREATE TABLE Tasks (  
    TaskID INT PRIMARY KEY,  
    Description TEXT,  
    RoomID INT,  
    FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID)  
    -- Другие атрибуты задачи  
);
```


SERVICEREQUESTS	
RequestID 	integer
Description	text
ClientID	integer

CLIENTS	
ClientID 	integer
Name	varchar(255)
ContactInfo	varchar(255)

RESERVATIONS	
ReservationID 	integer
CheckInDate	date
CheckOutDate	date
ClientID	integer
RoomID	integer

TASKS	
TaskID 	integer
Description	text
RoomID	integer

ROOMS	
RoomID 	integer
Type	varchar(255)
Cost	decimal(10, 2)



Современные механизмы безопасности в мобильных приложениях.



ChatGPT

Безопасность в мобильных приложениях - критически важный аспект, учитывая, что они часто содержат чувствительные данные пользователей. Вот несколько современных механизмов безопасности для мобильных приложений:

1. **Biometric Authentication (Биометрическая аутентификация):**
 - **Touch ID / Face ID (iOS) и Fingerprint API (Android):** Использование отпечатков пальцев или распознавания лица для аутентификации пользователей.
2. **OAuth и OpenID Connect:**
 - Использование протоколов аутентификации и авторизации, таких как OAuth и OpenID Connect, для безопасной аутентификации через сторонние сервисы.
3. **SSL/TLS для безопасной передачи данных:**
 - Обеспечение безопасной передачи данных между мобильным устройством и сервером с использованием протоколов шифрования SSL/TLS.
4. **Mobile Device Management (MDM):**
 - Использование MDM-решений для удаленного управления безопасностью мобильных устройств, включая блокировку, удаление данных и установку политик безопасности.
5. **App Transport Security (ATS):**
 - Включение ATS на iOS и Network Security Config на Android для обеспечения безопасности сетевого взаимодействия мобильного приложения.
6. **Runtime Application Self-Protection (RASP):**
 - Использование RASP-решений для обнаружения и предотвращения атак на уровне выполнения приложения.
7. **Многофакторная аутентификация (MFA):**
 - Внедрение многофакторной аутентификации для повышения уровня безопасности, включая комбинацию пароля, биометрических данных и одноразовых кодов.
8. **Безопасное хранение данных:**
 - Использование надежных механизмов для хранения чувствительных данных, таких как Secure Enclave на iOS и KeyStore на Android.
9. **Обнаружение и защита от вредоносного программного обеспечения:**
 - Использование антивирусных программ, обнаружение вредоносного программного обеспечения и регулярные обновления программного обеспечения для минимизации рисков.
10. **App Permissions и Privacy Controls:**
 - Тщательное управление разрешениями приложения для минимизации доступа к чувствительным данным и реализация мер контроля конфиденциальности.
11. **Обеспечение безопасности хранилища ключей:**
 - Защита ключей и других секретов с использованием хранилищ ключей (KeyStore на Android, Secure Enclave на iOS) для предотвращения их утечки.
12. **Penetration Testing и Code Review:**
 - Проведение тестов на проникновение и регулярных код-ревью для выявления потенциальных уязвимостей.

Эти механизмы безопасности являются лишь частью широкого спектра мер, которые можно принять для обеспечения безопасности мобильных приложений. Кроме того, важно следить за обновлениями и соблюдать передовые практики безопасности разработки.

Реализация аутентификации в мобильном приложении с помощью QR-кода.



ChatGPT

Аутентификация с использованием QR-кода в мобильном приложении предоставляет удобный и быстрый способ подтверждения личности пользователя. Ниже приведен пример шагов, которые могут быть реализованы для аутентификации с помощью QR-кода:

- 1. Создание QR-кода для каждого пользователя:**
 - Каждому пользователю генерируется уникальный QR-код, представляющий их идентификационные данные. Эти данные могут быть, например, уникальным идентификатором пользователя.
- 2. Интеграция QR-сканера в мобильное приложение:**
 - В мобильном приложении реализуется функциональность сканирования QR-кодов. Можно воспользоваться сторонними библиотеками для реализации QR-сканера, такими как ZXing для Android и iOS.
- 3. Сканирование QR-кода пользователем:**
 - Пользователь запускает мобильное приложение и выбирает опцию "Сканировать QR-код". Камера устройства активируется для сканирования.
- 4. Обработка данных после сканирования:**
 - После сканирования QR-кода, данные из него извлекаются и передаются в мобильное приложение. Эти данные могут быть зашифрованными, например, для обеспечения дополнительного уровня безопасности.
- 5. Отправка данных на сервер:**
 - Полученные данные отправляются на сервер для проверки. Сервер сопоставляет данные с базой данных, чтобы удостовериться в подлинности пользователя.
- 6. Генерация и отправка токена:**
 - Если пользователь успешно идентифицирован, сервер генерирует и возвращает токен аутентификации, который будет использоваться для последующих запросов.
- 7. Установка сессии:**
 - На стороне сервера устанавливается сессия для отслеживания действий пользователя после успешной аутентификации.
- 8. Обработка ошибок:**
 - Реализация обработки ошибок и безопасных механизмов для случаев, когда сканирование QR-кода не проходит успешно, или данные не могут быть верифицированы.
- 9. Тайм-аут сеанса:**
 - Установка временных рамок для сеанса аутентификации для предотвращения злоупотреблений.
- 10. Логирование событий:**
 - Регистрация событий аутентификации и проверка безопасности для последующего анализа и мониторинга.

Важно учесть, что безопасность играет ключевую роль при реализации аутентификации с использованием QR-кода, и следует уделять особое внимание защите передаваемых данных и противодействию возможным атакам.

Реализация в своем приложении сканера QR-кода.



ChatGPT

Для реализации сканера QR-кода в своем мобильном приложении, вы можете использовать сторонние библиотеки, предоставляющие функциональность сканирования. Вот пример использования библиотеки ZXing (Zebra Crossing), которая является популярной для сканирования QR-кодов. В этом примере будут представлены шаги для интеграции ZXing в Android-приложение с использованием Kotlin:

1. Добавьте зависимость в файл `build.gradle` для вашего приложения:

```
gradle Copy code

implementation 'com.google.zxing:core:3.4.1'
implementation 'com.journeyapps:zxing-android-embedded:4.2.0'
```

1. Добавьте разрешение на использование камеры в файл `AndroidManifest.xml`:

```
xml Copy code

<uses-permission android:name="android.permission.CAMERA" />
```

1. Создайте активити для сканирования QR-кода, например, `QRScannerActivity`:

```
kotlin Copy code

import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.zxing.integration.android.IntentIntegrator
import com.google.zxing.integration.android.IntentResult

class QRScannerActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_qr_scanner)

        // Инициализация сканера QR-кода
        val integrator = IntentIntegrator(this)
        integrator.setPrompt("Сканируйте QR-код") // Сообщение пользователю
        integrator.setOrientationLocked(false)
        integrator.setBeepEnabled(true) // Включение звукового сигнала при сканировании
        integrator.setCaptureActivity(CustomScannerActivity::class.java)

        // Запуск сканера
        integrator.initiateScan()
    }

    // Обработка результата сканирования
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        val result: IntentResult? = IntentIntegrator.parseActivityResult(requestCode, resultCode, data)
        if (result != null) {
            if (result.contents == null) {
                Toast.makeText(this, "Отменено", Toast.LENGTH_LONG).show()
            } else {
                Toast.makeText(this, "Сканирование успешно: ${result.contents}", Toast.LENGTH_LONG).show()
            }
        }
    }
}
```

```

    } else {
        Toast.makeText(this, "Сканирование успешно: ${result.content}", Toast.LENGTH_SHORT).show()
        // Здесь вы можете обработать полученные данные
    }
}
}
}
}

```

1. (Опционально) Создайте кастомную активити для сканирования (например, `CustomScannerActivity`), чтобы изменить внешний вид сканера:

```

kotlin
Copy code

import com.journeyapps.barcodescanner.CaptureActivity

class CustomScannerActivity : CaptureActivity()

```

1. Добавьте кнопку или другой элемент в ваш XML-макет для запуска `QRScannerActivity`.

Таким образом, приложение запускает `QRScannerActivity`, пользователь может использовать камеру устройства для сканирования QR-кода, и результат будет обработан в методе `onActivityResult`.

Современные методы шифрования конфиденциальных данных в базах данных.



ChatGPT

Шифрование данных в базах данных является ключевым элементом стратегии безопасности для защиты конфиденциальной информации. Ниже приведены несколько современных методов шифрования данных в базах данных:

1. Transparent Data Encryption (TDE):

- TDE предоставляет шифрование данных на уровне всей базы данных. Данные автоматически шифруются перед сохранением на диске и дешифруются при загрузке в память. Это обеспечивает прозрачное шифрование без необходимости изменения существующего приложения.

2. Column-level Encryption:

- Шифрование на уровне столбцов позволяет выбирать конкретные столбцы для шифрования. Это полезно, когда не все данные в базе данных требуют высокого уровня защиты, и вы можете выбирать, какие данные следует шифровать.

3. Application-layer Encryption:

- Вместо шифрования данных на уровне базы данных, шифрование может быть реализовано на уровне приложения. Приложение само шифрует данные перед их записью в базу данных и дешифрует их при извлечении. Это обеспечивает более гибкую и точечную защиту данных.

4. Homomorphic Encryption:

- Homomorphic Encryption предоставляет возможность выполнять вычисления над зашифрованными данными, не раскрывая ключ шифрования. Это позволяет обрабатывать данные в зашифрованной форме без необходимости расшифровывать их.

5. Tokenization:

- Вместо шифрования собственно данных, можно использовать токенизацию. Оригинальные данные заменяются случайными токенами, которые могут быть заменены обратно на исходные данные только с использованием ключа токенизации.

6. Key Management Systems (KMS):

- Эффективное управление ключами шифрования критически важно. KMS предоставляют инструменты для генерации, хранения и управления ключами шифрования.

7. Dynamic Data Masking:

- Динамическое маскирование данных (Dynamic Data Masking) предоставляет возможность скрывать часть данных для пользователей в зависимости от их роли или разрешений. Хотя это не шифрование, оно может служить дополнительным слоем защиты.

8. Secure Sockets Layer (SSL) / Transport Layer Security (TLS):

- Обеспечивает безопасную передачу данных между приложением и базой данных. Это важно для защиты данных в пути.

9. Multi-Party Computation (MPC):

- MPC позволяет нескольким сторонам выполнять вычисления над их данными, не раскрывая сами данные. Это особенно полезно в сценариях, где требуется совместное использование данных между несколькими участниками.

Выбор конкретного метода зависит от требований вашей системы, типа данных и контекста использования. Важно также правильно управлять ключами, регулярно обновлять шифрование и следовать передовым стандартам безопасности.

Реализация Application-layer Encryption в мобильном приложении.



ChatGPT

Реализация Application-layer Encryption в мобильном приложении подразумевает шифрование данных на уровне приложения перед сохранением их в базу данных и дешифрование при извлечении. Ниже представлен простой пример, демонстрирующий, как это можно сделать на платформе Android, используя язык программирования Kotlin и стандартные средства Android SDK.

Пример реализации Application-layer Encryption для простой базы данных в Android:

1. Добавьте зависимости:

В файле `build.gradle` вашего приложения добавьте зависимости:

```
gradle
Copy code

implementation 'androidx.room:room-runtime:2.4.0'
kapt 'androidx.room:room-compiler:2.4.0'
```

2. Создайте класс сущности данных:

```
kotlin
Copy code

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "encrypted_data")
data class EncryptedData(
    @PrimaryKey(autoGenerate = true) val id: Long = 0,
    @ColumnInfo(name = "encrypted_text") val encryptedText: String
)
```

3. Создайте класс базы данных с использованием Room:

```
kotlin
Copy code

import androidx.room.Database
import androidx.room.RoomDatabase

@Database(entities = [EncryptedData::class], version = 1, exportSchema = false)
abstract class AppDatabase : RoomDatabase() {
    abstract fun encryptedDataDao(): EncryptedDataDao
}
```

4. Создайте DAO (Data Access Object) для работы с базой данных:

```
kotlin
Copy code

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.Query

@Dao
interface EncryptedDataDao {
    @Insert
    fun insert(encryptedData: EncryptedData)

    @Query("SELECT * FROM encrypted_data")
    fun getAll(): List<EncryptedData>
}
```


5. Добавьте класс для управления шифрованием:

```
kotlin Copy code

import android.security.keystore.KeyGenParameterSpec
import android.security.keystore.KeyProperties
import java.security.KeyStore
import javax.crypto.Cipher
import javax.crypto.KeyGenerator
import javax.crypto.SecretKey

object EncryptionManager {
    private const val ANDROID_KEY_STORE = "AndroidKeyStore"
    private const val KEY_ALIAS = "MyKeyAlias"

    fun encryptData(data: String): String {
        val cipher = getCipher(Cipher.ENCRYPT_MODE)
        val encryptedBytes = cipher.doFinal(data.toByteArray())
        return android.util.Base64.encodeToString(encryptedBytes, andr
    }

    fun decryptData(encryptedData: String): String {
        val cipher = getCipher(Cipher.DECRYPT_MODE)
        val encryptedBytes = android.util.Base64.decode(encryptedData,
        val decryptedBytes = cipher.doFinal(encryptedBytes)
        return String(decryptedBytes)
    }

    private fun getCipher(mode: Int): Cipher {
        val keyStore = KeyStore.getInstance(ANDROID_KEY_STORE)
        keyStore.load(null)

        val key = keyStore.getKey(KEY_ALIAS, null) as SecretKey

        val cipher = Cipher.getInstance("${KeyProperties.KEY_ALGORITHM}
        cipher.init(mode, key)
        return cipher
    }

    fun generateKey() {
        val keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY_
        val keyGenSpec = KeyGenParameterSpec.Builder(KEY_ALIAS, KeyPro
            .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
            .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_PK
            .build()

        keyGenerator.init(keyGenSpec)
        keyGenerator.generateKey()
    }
}
```

В этом примере используется Android Keystore для генерации и хранения ключа, который используется для шифрования и дешифрования данных.

6. Используйте шифрование при сохранении и извлечении данных из базы данных:

```
kotlin Copy code

import android.content.Context
import androidx.room.Room

class DataManager(context: Context) {
    private val appDatabase: AppDatabase = Room.databaseBuilder(context

    init {
        EncryptionManager.generateKey() // Генерация ключа при первом запуске
    }

    fun saveEncryptedData(data: String) {
        val encryptedText = EncryptionManager.encryptData(data)
        val encryptedData = EncryptedData(encryptedText = encryptedText)
        appDatabase.encryptedDataDao().insert(encryptedData)
    }

    fun getDecryptedData(): List<String> {
        val encryptedDataList = appDatabase.encryptedDataDao().getAll()
        return encryptedDataList.map { EncryptionManager.decryptData(it) }
    }
}
```

В этом примере данные шифруются при сохранении и дешифруются при извлечении из базы данных.

Обратите внимание, что приведенный код служит исключительно для обучения и демонстрации концепций. Перед внедрением в реальное приложение, особенно если оно работает с конфиденциальными данными, необходимо тщательно обдумать безопасность и возможно проконсультироваться с экспертами по безопасности.

Методы анализа эффективности системы.



ChatGPT

Анализ эффективности системы является важной частью обслуживания и улучшения работы информационных систем. Вот несколько методов анализа эффективности системы:

1. Производительность системы:

- **Измерение времени ответа:** Оценка времени, требуемого системой на обработку запросов и предоставление результата.
- **Измерение загрузки ресурсов:** Мониторинг использования CPU, памяти, диска и сети для выявления узких мест и бутылочных горлышек.

2. Надежность и устойчивость:

- **Анализ отказов и ошибок:** Изучение журналов событий и отчетов об ошибках для выявления проблем и улучшения стабильности системы.
- **Мониторинг времени безотказной работы (MTBF) и времени восстановления (MTTR):** Определение среднего времени между отказами и среднего времени восстановления после отказа.

3. Масштабируемость и производительность:

- **Тестирование масштабируемости:** Проведение нагрузочных тестов для определения, как система справляется с ростом числа пользователей и объема данных.
- **Оценка производительности при добавлении ресурсов:** Изучение того, как система ведет себя при увеличении выделенных ресурсов.

4. Удовлетворенность пользователей:

- **Обратная связь пользователей:** Сбор обратной связи от пользователей с использованием анкет, опросов и комментариев для определения их удовлетворенности и выявления возможных улучшений.
- **Анализ пользовательского опыта (UX):** Оценка удобства использования, доступности и общего восприятия системы.

5. Соблюдение требований безопасности:

- **Аудит безопасности:** Проведение аудита системы для выявления потенциальных уязвимостей и обеспечения соблюдения стандартов безопасности.
- **Мониторинг активности:** Отслеживание событий безопасности и анализ журналов для выявления подозрительной активности.

6. Анализ использования ресурсов:

- **Оптимизация запросов к базе данных:** Проверка и улучшение запросов к базе данных для оптимизации производительности.
- **Мониторинг использования сети:** Анализ сетевого трафика для выявления узких мест и оптимизации передачи данных.

7. Анализ данных о состоянии системы:

- **Использование метрик и KPI:** Определение ключевых показателей эффективности (KPI) и метрик для оценки работы системы.
- **Построение графиков и дашбордов:** Визуализация данных для облегчения анализа и принятия решений.

8. Согласованность с бизнес-целями:

- **Оценка соответствия целям бизнеса:** Анализ того, насколько система способствует достижению целей организации.

Эти методы могут использоваться в комбинации для обеспечения всестороннего анализа эффективности системы. Важно регулярно проводить мониторинг и аудит, чтобы оперативно выявлять проблемы и внедрять улучшения.

Сгенерируем фавикон с помощью Midjourney.



Заключение

В ходе работы были приобретены навыки работы с большой языковой моделью Chat-GPT версии 3.5 и нейросетью-художником Midjourney, полученные результаты считаю полезными, и они определенно пригодятся при подготовке к выпускной квалификационной работе.