Из Linux в Android

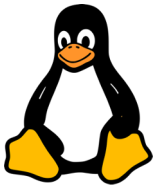
Путь платежного терминала

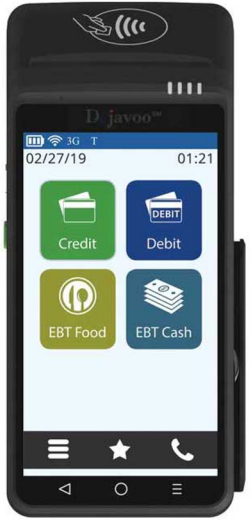
Малюгин Платон

Лид Android разработки в Dejavoo Systems Russia

Задача


Dejavoo Z11
03/19/18
11:05
Credit
Debit
EBT Food
EBT Cash
OK




Dejavoo
02/27/19
01:21
Credit
Debit
EBT Food
EBT Cash

Мотивы перехода на Android

Низкая цена

Мощность

Sexy

Поддержка

Какое у нас ТЗ?

Сделайте как в платежном терминале на Linux

Секции

- Как происходит платеж
- Переход на Android
- Разработка
- Тестирование

Как происходит платеж*

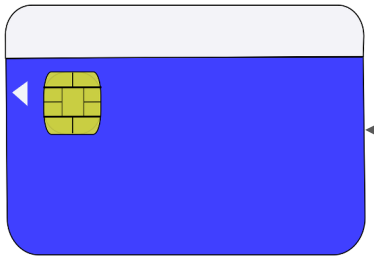
*транзакция

Участники платежа

EMV карта
(карта)

Терминал

Платежные хосты
(хосты)



EMV Kernel

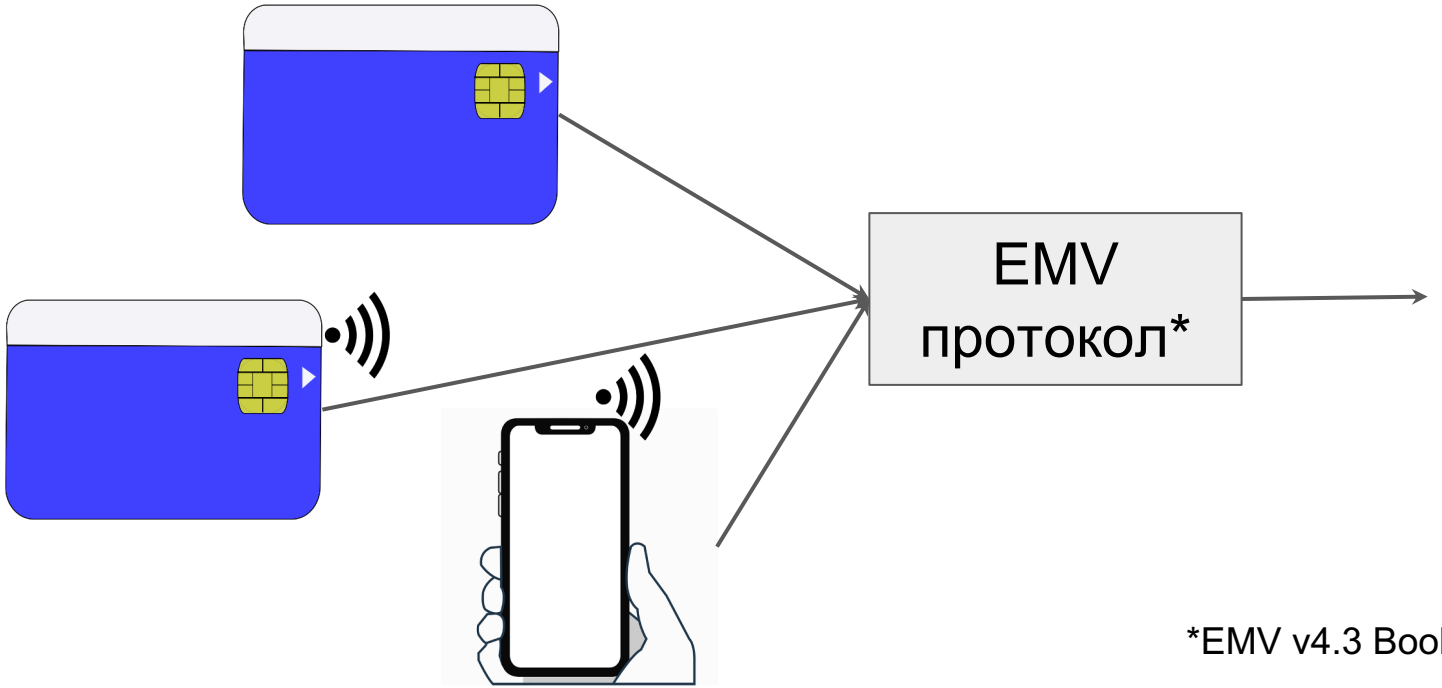
Спецификация

EMVCo

Платеж


Питание
Данные
Карт ридер
Вставили карту
Создается сессия

Платеж

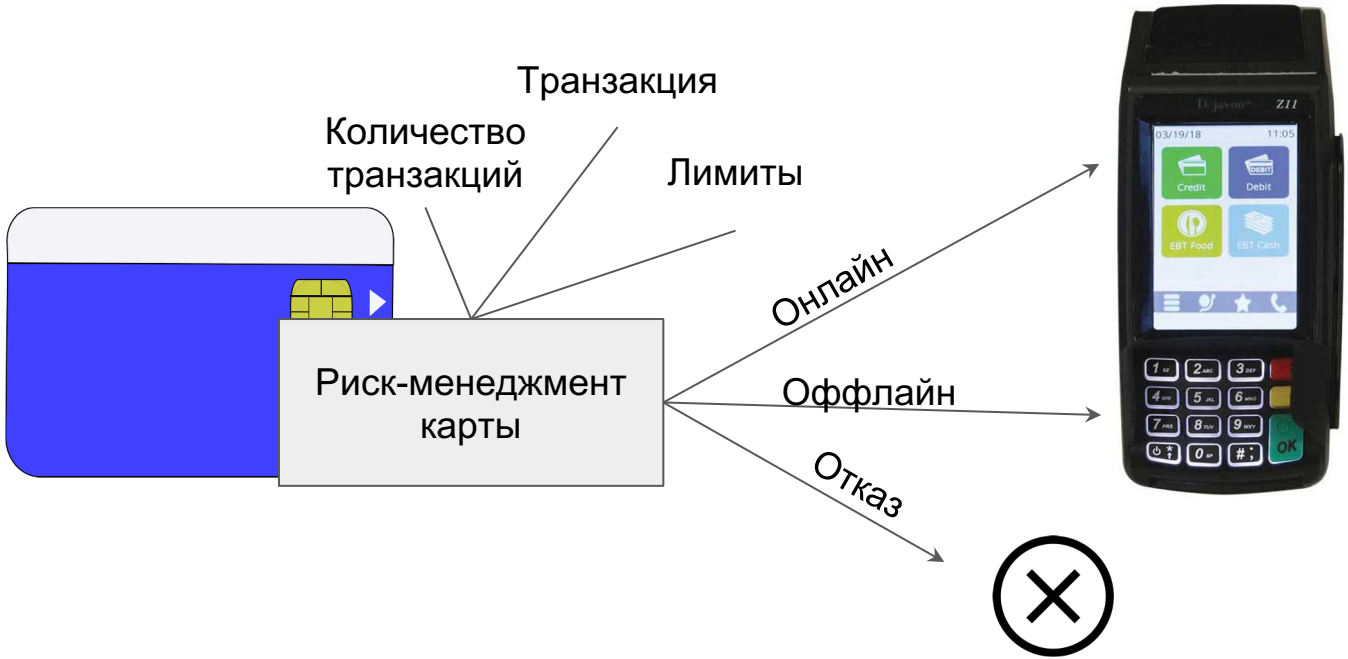


*EMV v4.3 Book 3

Платеж


Транзакция
Вставили карту
Риск-менеджмент терминала
Онлайн
Оффлайн
Отказ
Конфигурация

Платеж

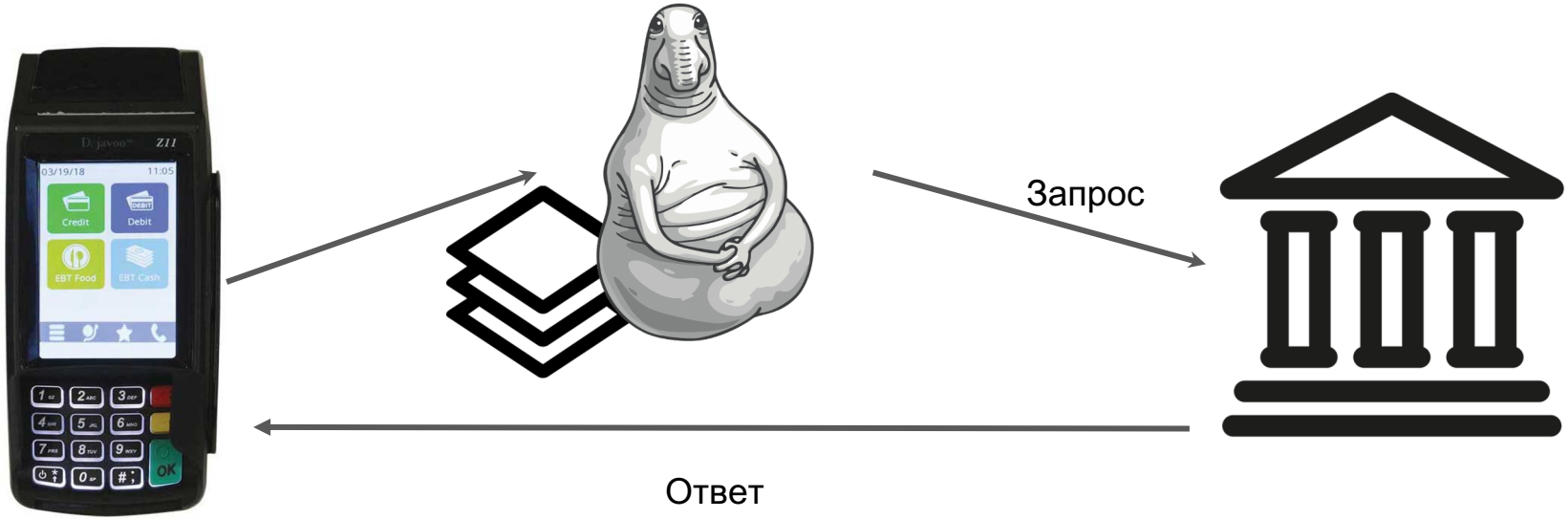

Транзакция
Количество транзакций
Лимиты
Риск-менеджмент карты
Онлайн
Оффлайн
Отказ

Платеж (Онлайн)

Криптопрограмма (ARQC)

Криптопрограмма (ARPC)

Платеж (Оффлайн)

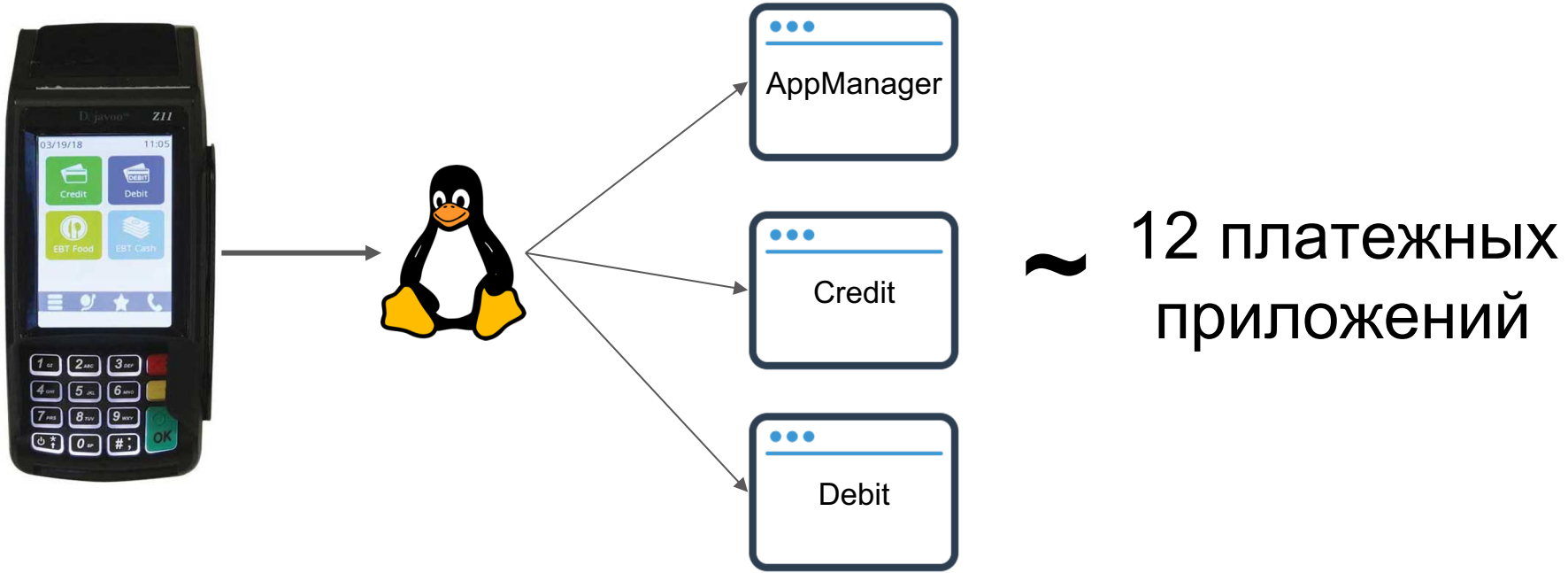


Подытожим

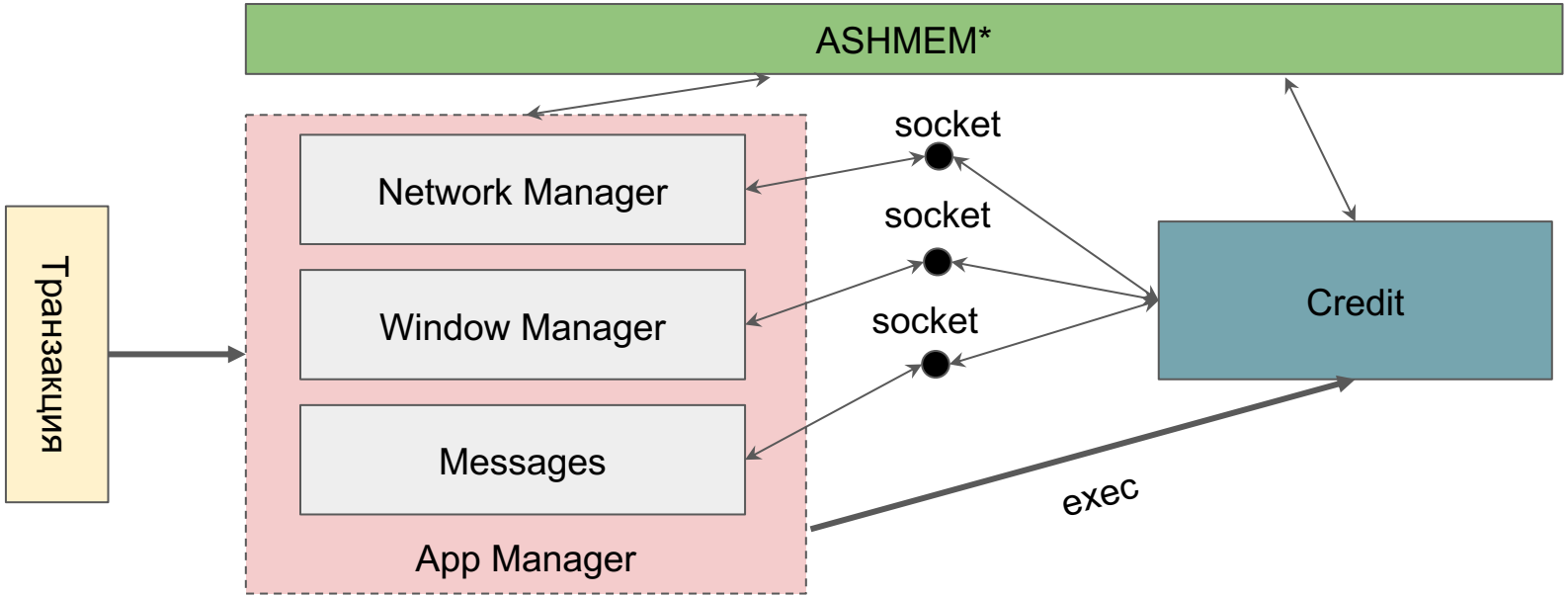
- При платеже минимум три участника
- Риск-менеджмент влияет на платеж, у карты приоритет
- Оплата может быть Online и Offline
- Один EMV протокол, различается только транспорт

Переход на Android

С чего начали

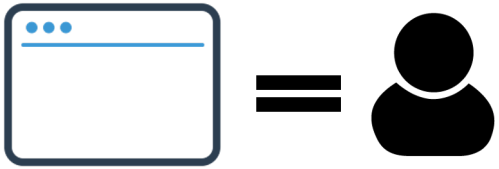


Как это все работает



*Anonymous shared memory

Разница с Linux



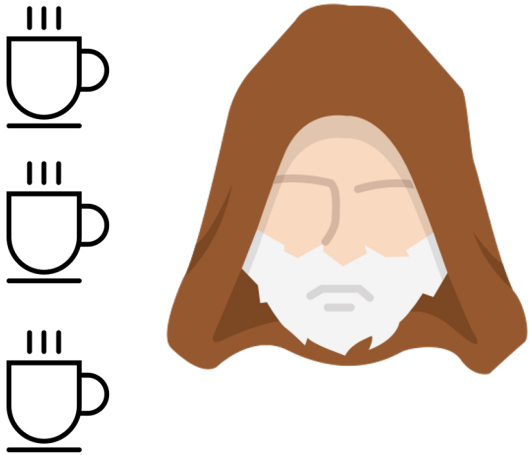
Android UI

~~ASHMEM*~~ ➡ MMAP

Сокеты

* добавили в API26

На чьей ты стороне

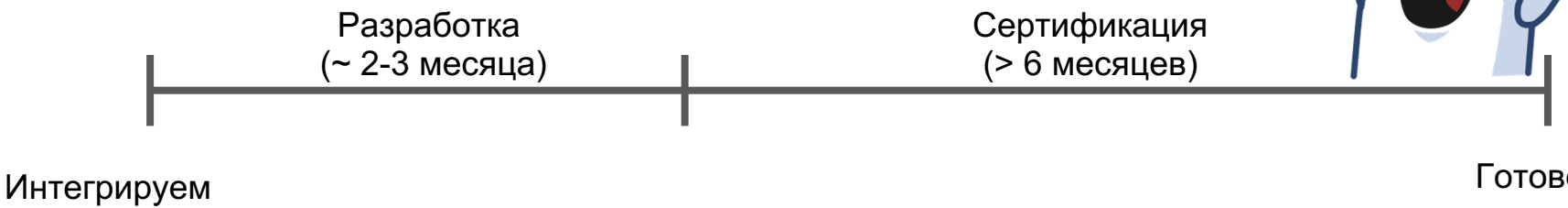


Переписать

Переиспользовать

Почему я выбрал темную сторону?

- Все уже написано и отлажено, не нужно копировать код
- Сможем поддерживать две версии Android и Linux
- Сможем сразу сертифицировать хост под две платформы



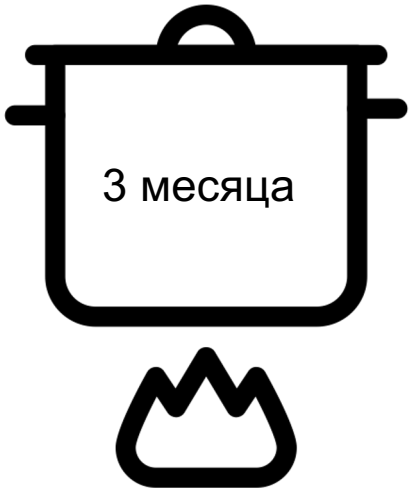
Среда для сертификации открывается на пару дней

Как готовим кросс-платформу

C++

OpenSSL
Cryptography and SSL/TLS Toolkit

curl://



6.0

JNI

Эволюция приложения

Ядро управляет всем

Стэком

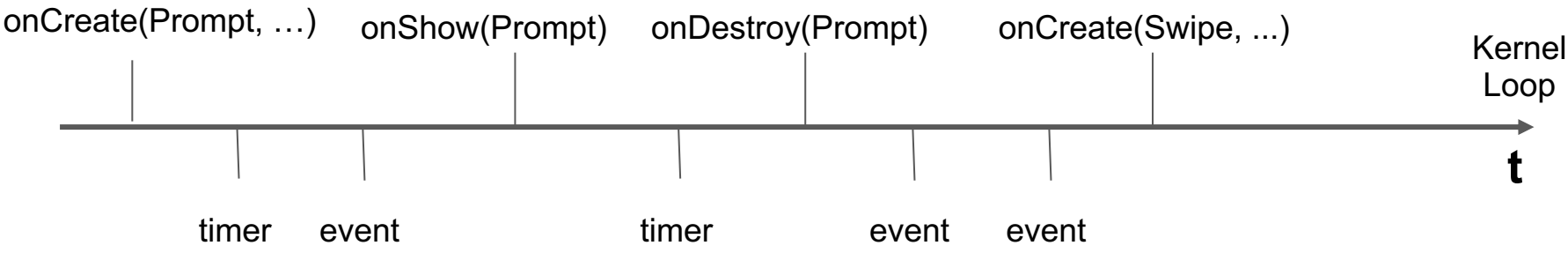
Экранами

Локализацией

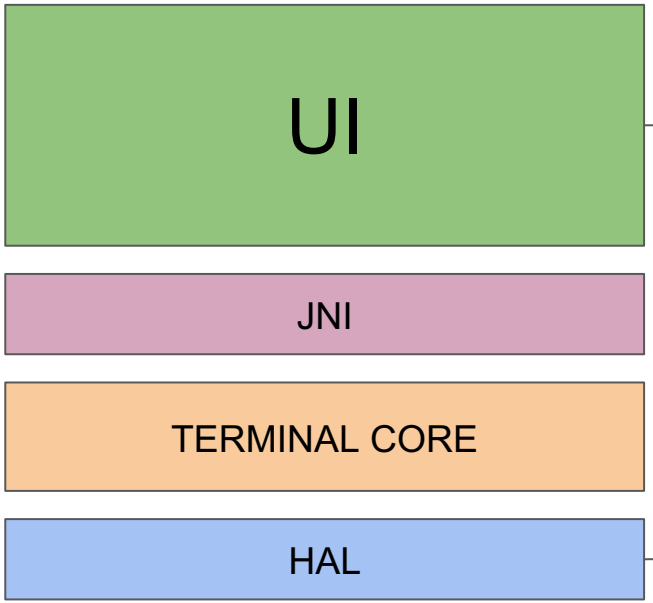
Настройками

События/Таймерами

Kernel Loop ядра



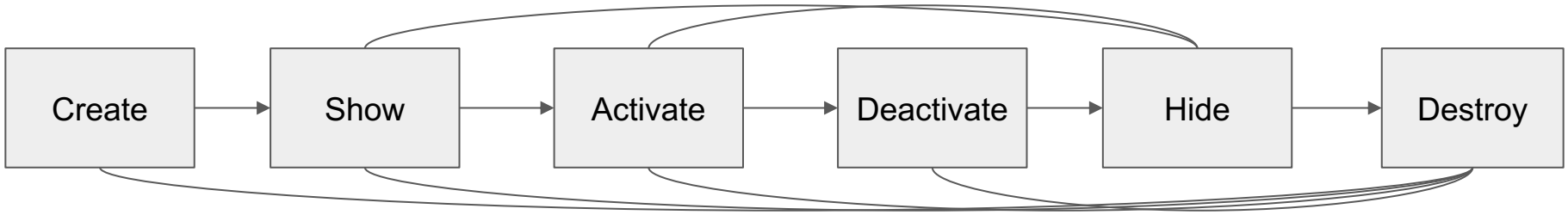
Что в итоге получилось



- Single Activity
- Экран = ?
- Поддержка анимации

Новая платформа

Жизненный цикл экрана



Activity vs **Fragment** vs **View**

Почему View для экранов

Кастомная реализация стэка

Кастомный жизненный цикл экрана

Синхронное отображение

UI фреймворк

```
Canvas c;

BufferCanvas bc;
bc.set(MAX_WIDTH, 80);
bc.fill(0, 0, MAX_WIDTH, 80, 0);
ClipRgn clip = { bc.buffer(), 0, 0 };

c.intersectClipRgn(clip);
```

Примитивы

```
<Button
  android:layout_width="match_parent"
  android:layout_height="80dp"
  android:text="Test" />
```

XML разметка

Управление стэком из ядра

```
void handleDestroy(String tag);
void handleActivate(String tag);
void handleDeactivate(String tag);
void handleShow(String tag);
void handleHide(String tag);
```

Java

А где Create?

Как выглядит типичный экран. Kotlin

```kotlin
class AlertScreen(
  private val caption: String,
  private val message: String,
  private val buttons: List<String>,
  private val selectedIndex: Int
) : Screen {
  override val tag: String = "Alert"
  override fun createView(context: Context): WindowFragment = AlertWindowFragment(
    caption, message, buttons, selectedIndex
  )
}
```

Как выглядит типичный экран. C++

```
class AlertDialogWindow : public Window {
  public:
    ModalDialogWindow(const char *title) : _title(title), Window("Alert") {
       CONNECT(timer_.onTimeout, handleInputTimeout, this);
    }
  protected:
    void processCreate(); // override
  private:
    const char *_title;
    CallbackTimer timer_;
    void handleInputTimeout(Timer&) { endModal(-1); }
  };
}
```

Подытожим

- Нет больших проблем при переходе с Linux на Android, всегда есть альтернатива
- Для UI используем Single Activity, где каждый экран View
- View отлично подходит когда необходимая кастомное управление стэком и кастомный жизненный цикл

Разработка

Legacy кода

- Был “заточен” под определенную платформу
- Может “выстрелить” в любой момент

Решение:

- Пометить все проблемные места (делать по частям)
- Добавить больше логов
- Ждать =(

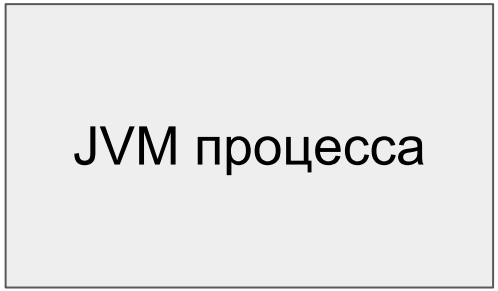
“Подружить” лупы

```
blocker = CountDownLatch(1)
handler.post {
  callback() // UI Thread
  blocker.countDown()
}
try {
  blocker.await()
} catch (e: InterruptedException) { }
```

Как дернуть Java/Kotlin метод из C++

Шаг 1. Получить ссылку на JVM

```
extern "C"
JNIEXPORT void JNICALL
Java_aura_Window_init(JNIEnv *env, jobject instance) {
  jint res = env->GetJavaVM(&g_JavaVM);
  if (res != JNI_OK) {
    // fatal
   }
}
```



Как дернуть Java/Kotlin метод из C++

Шаг 2. Добавить ссылку на экземпляр класса

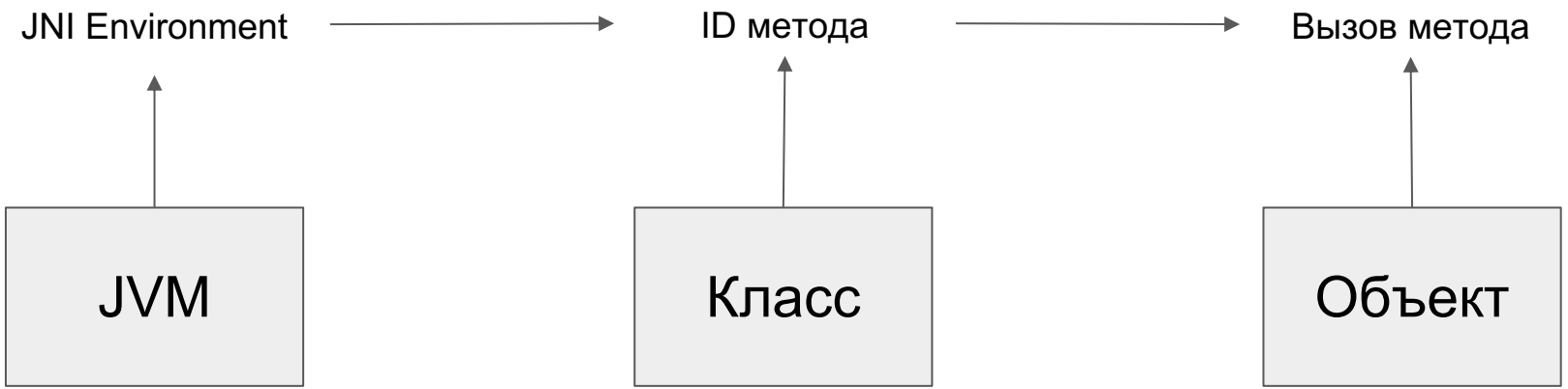
```
extern "C"
JNIEXPORT void JNICALL
Java_aura_Window_init(JNIEnv *env, jobject instance) {
  // ….
  jclass localClass = env->FindClass("aura/Window");
  if (env->ExceptionCheck()) { /* fatal */ }
  g_GlobalClass = (jclass) env->NewGlobalRef(localClass);
  if (env->ExceptionCheck()) {  /* fatal */  }
}
```

Как дернуть Java/Kotlin метод из C++

Шаг 3. Добавить ссылку на текущую сущность

```
extern "C"
JNIEXPORT void JNICALL
Java_aura_Window_init(JNIEnv *env, jobject instance) {
  // ….
  g_GlobalObject = (jobject) env->NewGlobalRef(instance);
  if (env->ExceptionCheck()) {  /* fatal */  }
}
```

Как дернуть Java/Kotlin метод из C++



Как дернуть Java/Kotlin метод из C++

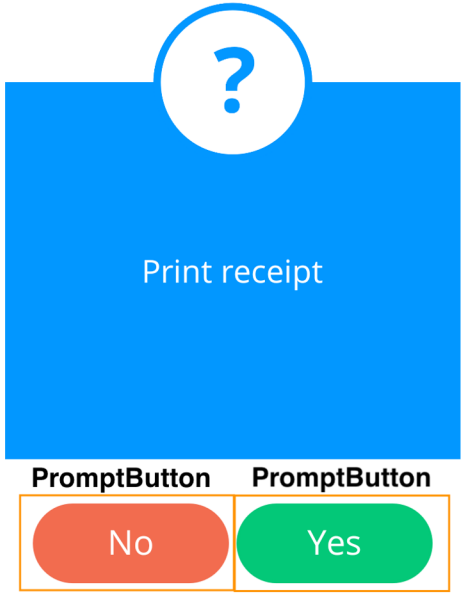
```
JNIEnv *env;
jint rs = g_JavaVM->AttachCurrentThread(&env, NULL);
if (rs != JNI_OK) { /* fatal */ }

jmethodID methodId = env->GetMethodID(g_GlobalClass,"open","()V");
if (env->ExceptionCheck()) { /* fatal */ }

env->CallVoidMethod(g_GlobalObject, methodId);
```

Зачем так много глобальных переменных?

Связь моделей Java и C++




```java
public class PromptButton {
  private final byte[] name;
  private final int key;
  public PromptButton(byte[] name, int key) {
    this.name = name;
    this.key = key;
  }
  public String getNameAsString() {
    return new ACDString(name).asString();
  }
  public int getKey() { return key; }
}
```

```cpp
class JPromptButton {
public:
    JPromptButton(const char *name, int key)
          : _name(name), _key(key) {}

    jobject asJavaObject(JNIEnv *env);

private:
    const char *_name;
    int _key;
};
```

C++

Подготавливаем Java объект

```
jobject JPromptButton::asJavaObject(JNIEnv *env) {
  jclass jClass = env->FindClass("aura/PromptButton");
  jmethodID jConstructorMethod = env->GetMethodID(jClass, "<init>", "([BI)V");

  return env->NewObject(
    jClass,
    jConstructorMethod,
    JavaStringCompat(_name).asJavaByteArray(env),
    (jint) _key
  );
}
```

Зачем такие сложности

```
env->GetMethodID(
  g_GlobalClass,
  "DisplayButton",
  "(Laura/PromptButton;)V"
);
```

C++

```
void DisplayButton(PromptButton button)
{
  // code
}
```

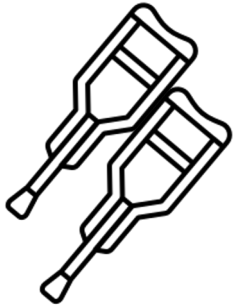
Java

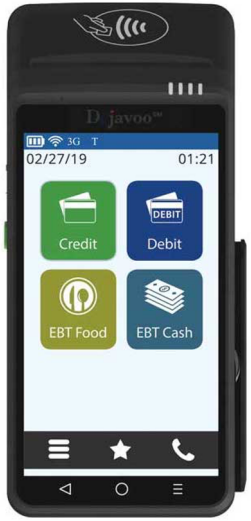
Переезд HAL-а

HAL

Извлекли Header

Vendor SDK





HAL

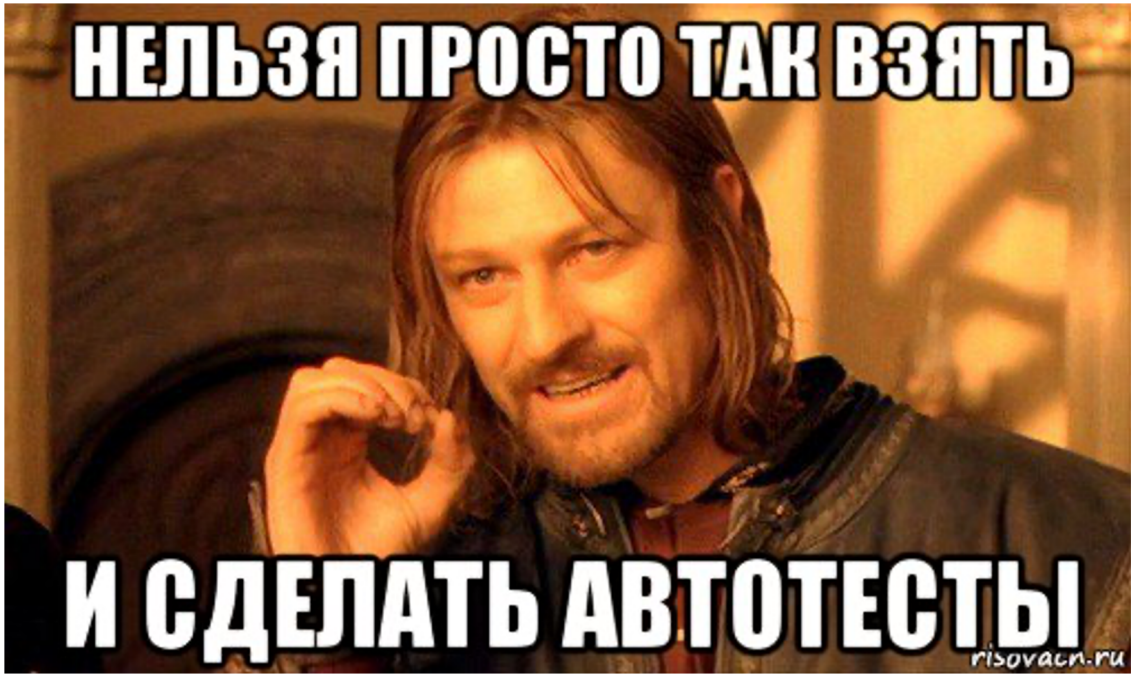
Подытожим

- Legacy вызывает проблему, но кросс-платформа определенно делает код чище
- Java/Kotlin и C/C++ это два разных мира
- Получилось минимальными усилиями, с костылями и палками, получить новый HAL

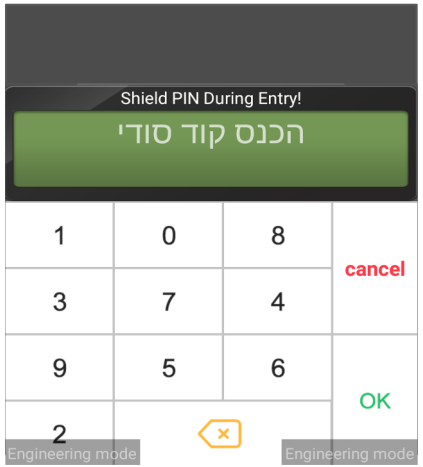
Но это работает

Тестирование

Тестирование



Есть взаимодействие с пользователем

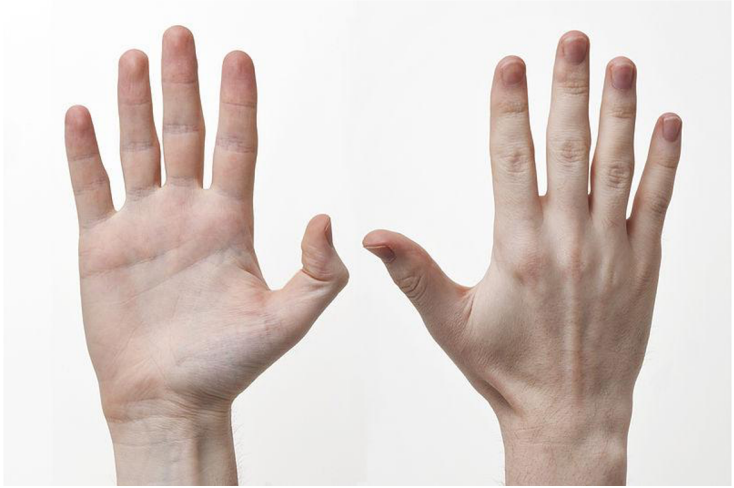


Не получится просто эмулировать карту

Не замокаешь все платежные хосты

- Огромные спецификации
- У хоста есть тестовая среда
- Слишком много работы, так как таких хостов больше 20

И как мы тестируем



+

А как хочется

https://www.abrantix.com/AXRobot.html

Итого

- Мигрировать с Linux на Android возможно малой кровью
- Кросс-платформа делает код чище
- Java и C++ это два разных мира, но их можно “подружить”
- Можно мигрировать на любой Android девайс*

* удовлетворяет требованиям платежной индустрии

Спасибо
Вопросы?

maluginp

maluginp