ОС Аврора 5 глазами мобильного разработчика

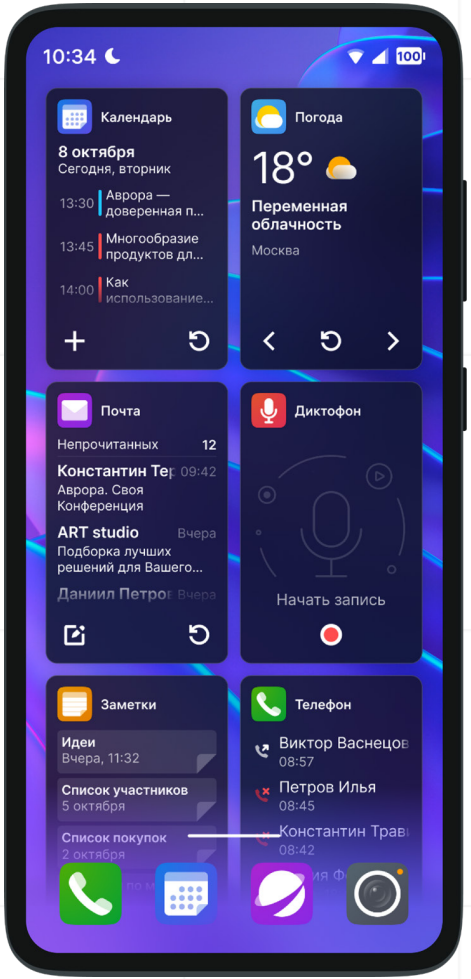
ОТКРЫТАЯ МОБИЛЬНАЯ ПЛАТФОРМА

Разработка и функционал

Глазков Денис
Старший инженер-разработчик отдела разработки ОС

Отечественная мобильная операционная система

Современный мобильный функционал с фокусом на безопасность и эргономику



Растущая экосистема приложений

Более 100 партнеров создают свои решения для ОС Аврора

Поддержка более 10 устройств

Смартфоны и планшеты

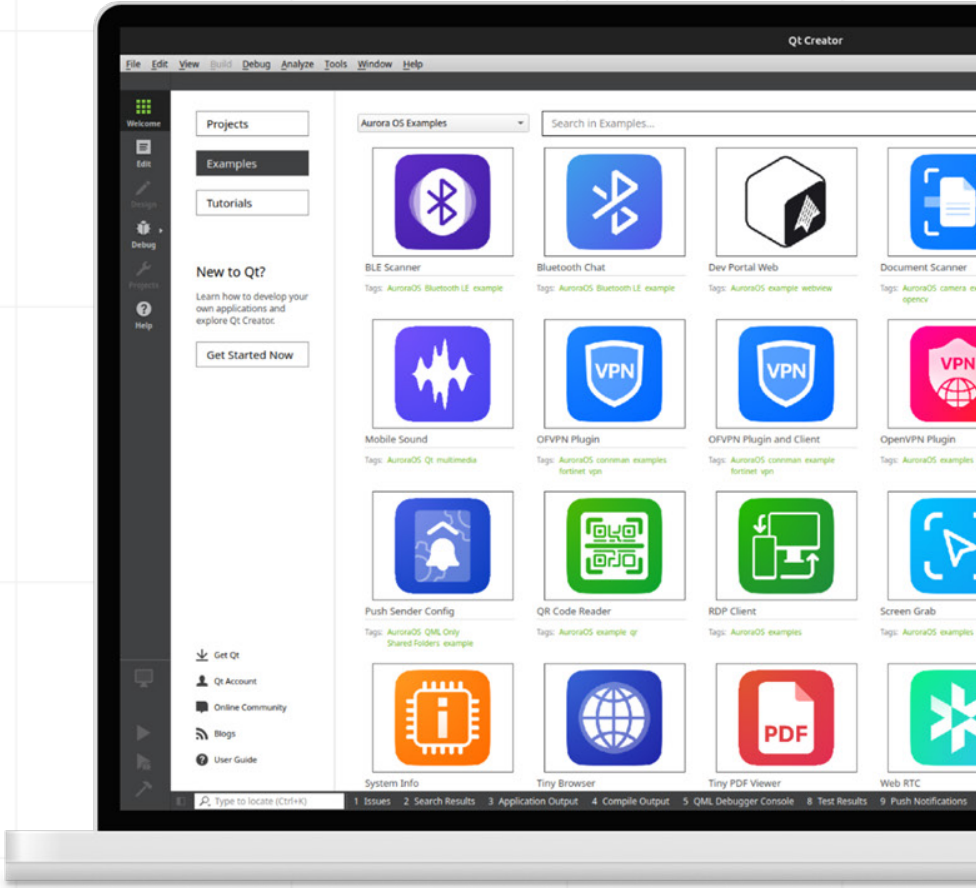
Более 500 000 устройств

В промышленной эксплуатации

Аврора SDK

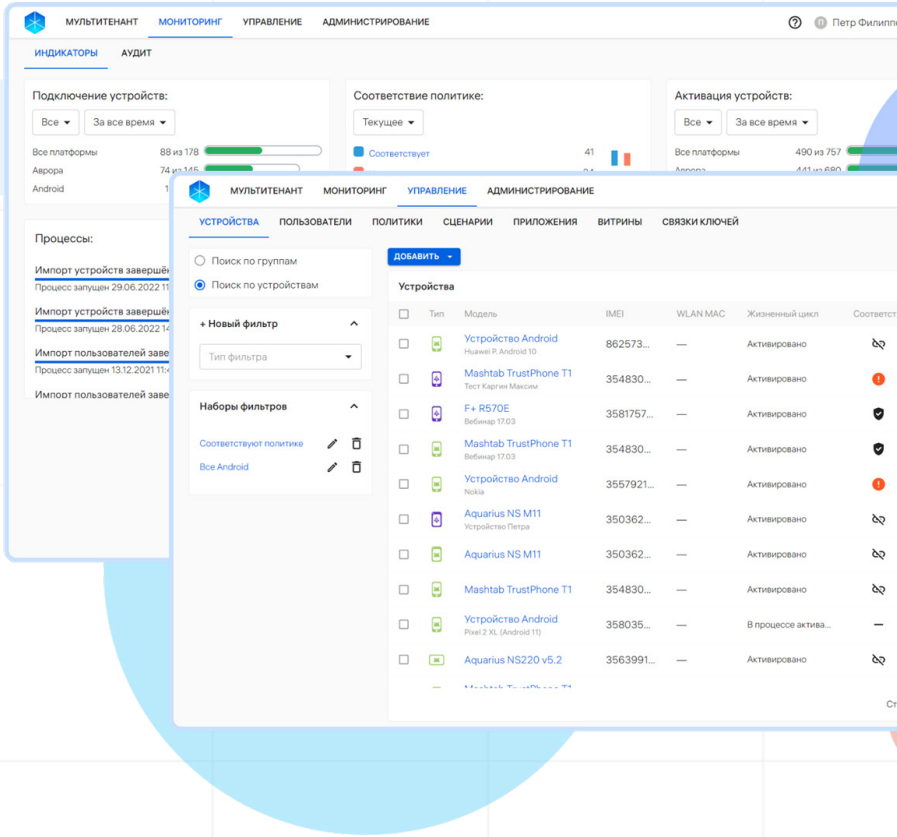
Удобная среда разработки под ОС Аврора

- Работа с зависимостями
- Эмулятор
- Кроссплатформенная сборка
- Утилиты для разработчика
- Большое количество примеров



Аврора Центр

- Управление парком устройств
- Push-сервис
- Сервис обновлений
- Магазин приложений



- Дочерняя компания ПАО «Ростелеком»
- Является участником open-source проектов и организаций
- Ведет образовательные программы в ведущих ВУЗах страны
- Офисы разработки в Москве, Санкт-Петербурге, Нижнем Новгороде и Иннополисе

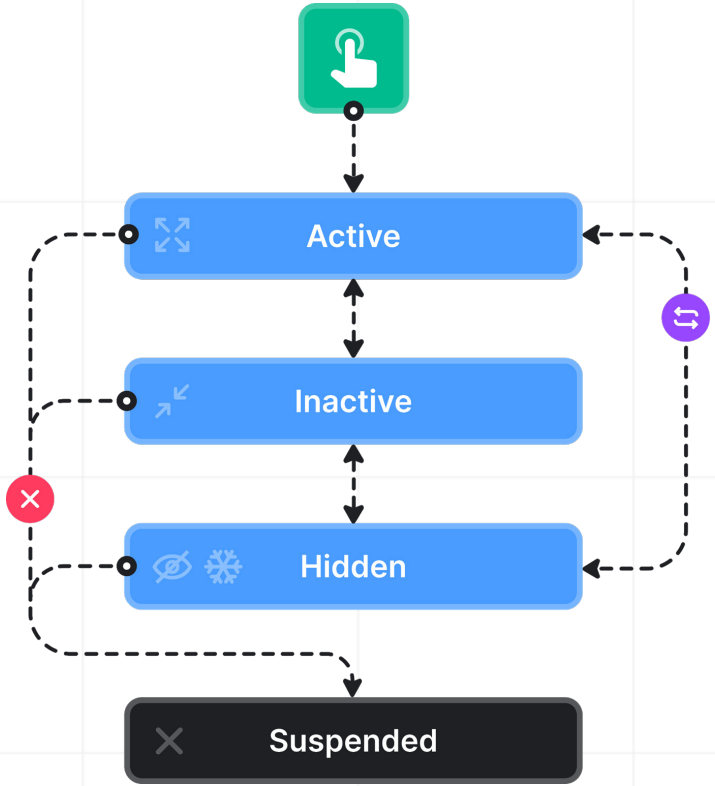
Нововведения в ОС Аврора 5.0

Runtime Manager

Runtime Manager

Управление жизненным циклом приложений

Жизненный цикл приложения в Аврора 5



Обработка изменения состояния приложения

```
QGuiApplication *app = Aurora::Application::application(argc, argv);

QObject::connect(app, QGuiApplication::applicationStateChanged,
                 [](Qt::ApplicationState state) {
    switch (state) {
        case Qt::ApplicationActive:
            qDebug() << "Application focused";
            break;
        case Qt::ApplicationInactive:
            qDebug() << "Application cover visible";
            break;
        case Qt::ApplicationHidden:
            qDebug() << "Application not visible";
            break;
        case Qt::ApplicationSuspended:
            qDebug() << "Application is about to be teminated";
            break;
        }
    });
```

Обработка изменения состояния приложения

```
QGuiApplication *app = Aurora::Application::application(argc, argv);
```

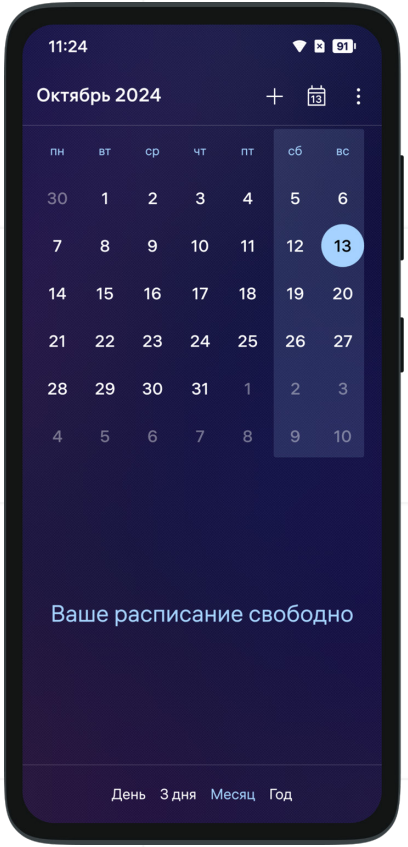
```
QObject::connect(app, QGuiApplication::applicationStateChanged,
                 [](Qt::ApplicationState state) {
    switch (state) {
        case Qt::ApplicationActive:
            qDebug() << "Application focused";
            break;
        case Qt::ApplicationInactive:
            qDebug() << "Application cover visible";
            break;
        case Qt::ApplicationHidden:
            qDebug() << "Application not visible";
            break;
        case Qt::ApplicationSuspended:
            qDebug() << "Application is about to be teminated";
            break;
        }
    });
```

Обработка изменения состояния приложения

```
QGuiApplication *app = Aurora::Application::application(argc, argv);

QObject::connect(app, QGuiApplication::applicationStateChanged,
                 [](Qt::ApplicationState state) {
    switch (state) {
        case Qt::ApplicationActive:
            qDebug() << "Application focused";
            break;
        case Qt::ApplicationInactive:
            qDebug() << "Application cover visible";
            break;
        case Qt::ApplicationHidden:
            qDebug() << "Application not visible";
            break;
        case Qt::ApplicationSuspended:
            qDebug() << "Application is about to be teminated";
            break;
        }
    });
```

Обработка изменения состояния приложения

```
QGuiApplication *app = Aurora::Application::application(argc, argv);

QObject::connect(app, QGuiApplication::applicationStateChanged,
                 [](Qt::ApplicationState state) {
    switch (state) {
        case Qt::ApplicationActive:
            qDebug() << "Application focused";
            break;
        case Qt::ApplicationInactive:
            qDebug() << "Application cover visible";
            break;
        case Qt::ApplicationHidden:
            qDebug() << "Application not visible";
            break;
        case Qt::ApplicationSuspended:
            qDebug() << "Application is about to be teminated";
            break;
        }
    });
```

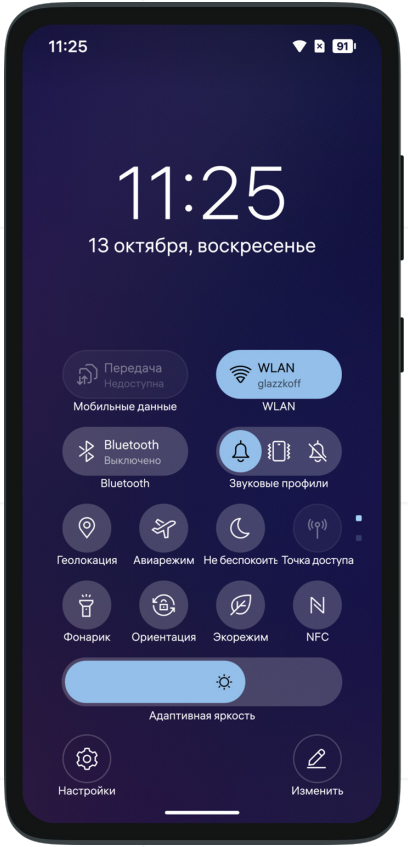


Обработка изменения состояния приложения

```
QGuiApplication *app = Aurora::Application::application(argc, argv);

QObject::connect(app, QGuiApplication::applicationStateChanged,
                 [](Qt::ApplicationState state) {
    switch (state) {
        case Qt::ApplicationActive:
            qDebug() << "Application focused";
            break;
        case Qt::ApplicationInactive:
            qDebug() << "Application cover visible";
            break;
        case Qt::ApplicationHidden:
            qDebug() << "Application not visible";
            break;
        case Qt::ApplicationSuspended:
            qDebug() << "Application is about to be teminated";
            break;
        }
    });
```

Обработка изменения состояния приложения

```
QGuiApplication *app = Aurora::Application::application(argc, argv);

QObject::connect(app, QGuiApplication::applicationStateChanged,
                 [](Qt::ApplicationState state) {
    switch (state) {
        case Qt::ApplicationActive:
            qDebug() << "Application focused";
            break;
        case Qt::ApplicationInactive:
            qDebug() << "Application cover visible";
            break;
        case Qt::ApplicationHidden:
            qDebug() << "Application not visible";
            break;
        case Qt::ApplicationSuspended:
            qDebug() << "Application is about to be teminated";
            break;
        }
    });
```



Обработка изменения состояния приложения

```
QGuiApplication *app = Aurora::Application::application(argc, argv);

QObject::connect(app, QGuiApplication::applicationStateChanged,
                 [](Qt::ApplicationState state) {
    switch (state) {
        case Qt::ApplicationActive:
            qDebug() << "Application focused";
            break;
        case Qt::ApplicationInactive:
            qDebug() << "Application cover visible";
            break;
        case Qt::ApplicationHidden:
            qDebug() << "Application not visible";
            break;
        case Qt::ApplicationSuspended:
            qDebug() << "Application is about to be teminated";
            break;
        }
    });
```

Обработка изменения состояния приложения

```
QGuiApplication *app = Aurora::Application::application(argc, argv);

QObject::connect(app, QGuiApplication::applicationStateChanged,
                 [](Qt::ApplicationState state) {
    switch (state) {
        case Qt::ApplicationActive:
            qDebug() << "Application focused";
            break;
        case Qt::ApplicationInactive:
            qDebug() << "Application cover visible";
            break;
        case Qt::ApplicationHidden:
            qDebug() << "Application not visible";
            break;
        case Qt::ApplicationSuspended:
            qDebug() << "Application is about to be teminated";
            break;
        }
    });
```

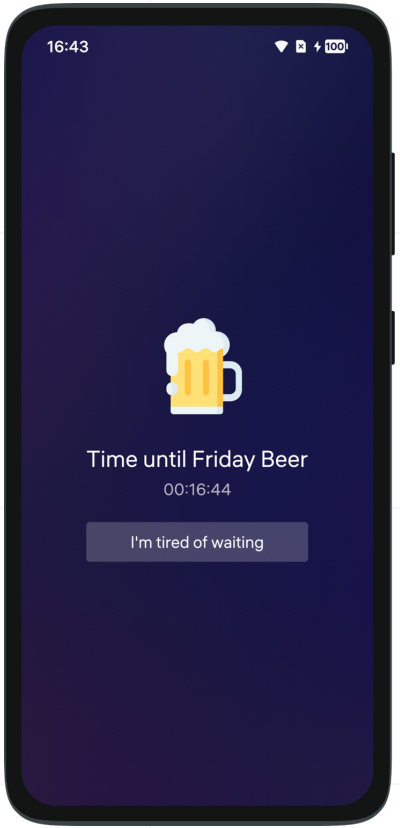
Runtime Manager

API для фоновых задач и интентов

Пример приложения, использующего Runtime Manager API

Beer Reminder

GitHub



Desktop файл приложения для ОС Аврора

Аналог манифеста в ОС Android

```
[Desktop Entry]
Type=Application
Exec=/usr/bin/dev.glazkov.BeerReminder
Icon=dev.glazkov.BeerReminder
Name=Friday Beer

[X-Application]
OrganizationName=dev.glazkov
ApplicationName=BeerReminder
```

Объявление фоновой задачи в desktop файле

```
[X-Task Remind]
Type=scheduled
Conditions=internet
Permissions=
```

Документация

Объявление фоновой задачи в desktop файле

Имя фоновой задачи

```
[X-Task Remind]
Type=scheduled
Conditions=internet
Permissions=
```

Объявление фоновой задачи в desktop файле

Тип фоновой задачи

```
[X-Task Remind]
Type=scheduled
Conditions=internet
Permissions=
```

- scheduled

  Фоновая задача, которая запускается в определенные моменты времени

- periodic

  Фоновая задача, которая запускается через определенный момент времени

- worker

  Фоновая задача, которая запускается по инициативе приложения

- session-start

  Фоновая задача, которая запускается при запуске устройства

Объявление фоновой задачи в desktop файле

Условия фоновой задачи

```
[X-Task Remind]
Type=scheduled
Conditions=internet
Permissions=
```

- battery-not-low

  Уровень зарядки не низкий
- charging

  Устройство заряжается
- internet

  Имеется доступ в интернет
- idle

  Устройство простаивает (Дисплей выключен)

Объявление фоновой задачи в desktop файле

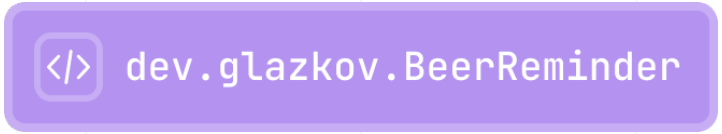
Разрешения фоновой задачи

```
[X-Task Remind]
Type=scheduled
Conditions=internet
Permissions=
```

Разрешения
ОС Аврора

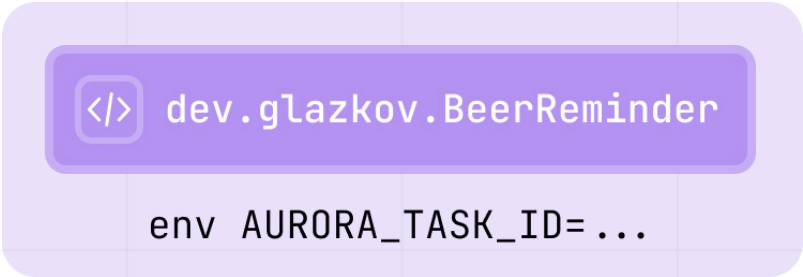
Процесс запуска фоновой задачи

Фоновая задача запускается при помощи главного исполняемого файла приложения



Процесс запуска фоновой задачи

При запуске фоновой задачи Runtime Manager добавляет в процесс переменную окружения с именем фоновой задачи



Разделение кода приложения и кода фоновой задачи

Подключаем библиотеку Runtime Manager

```
#include <RuntimeManager/RuntimeDispatcher>

int main(int argc, char *argv[])
{
    QGuiApplication *app = Aurora::Application::application(argc, argv);
    RuntimeDispatcher *dispatcher = RuntimeDispatcher::instance();

    dispatcher->onTaskStarted("Remind", [](const QString &backgroundTaskID) {
        remindAboutBeer();
    });

    dispatcher->onApplicationStarted([] {
        QQuickView *view = Aurora::Application::createView();
        view->setSource(Aurora::Application::pathTo("qml/dev.glazkov.BeerReminder.qml"));
        view->show();

        initRemindBackgroundTask();
    });

    return app->exec();
}
```

Разделение кода приложения и кода фоновой задачи

Получаем диспетчер исполнения

```
#include <RuntimeManager/RuntimeDispatcher>

int main(int argc, char *argv[])
{
    QGuiApplication *app = Aurora::Application::application(argc, argv);
    RuntimeDispatcher *dispatcher = RuntimeDispatcher::instance();

    dispatcher->onTaskStarted("Remind", [](const QString &backgroundTaskID) {
        remindAboutBeer();
    });

    dispatcher->onApplicationStarted([] {
        QQuickView *view = Aurora::Application::createView();
        view->setSource(Aurora::Application::pathTo("qml/dev.glazkov.BeerReminder.qml"));
        view->show();

        initRemindBackgroundTask();
    });

    return app->exec();
}
```

Разделение кода приложения и кода фоновой задачи

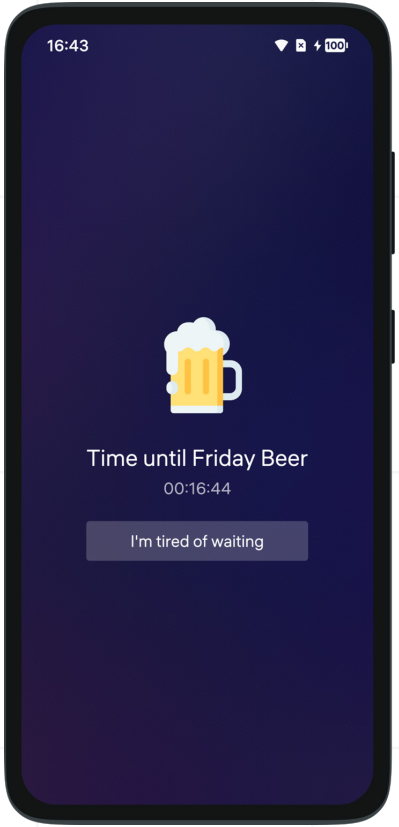
Указываем блок кода, который будет выполнен только в фоновой задаче

```cpp
#include <RuntimeManager/RuntimeDispatcher>

int main(int argc, char *argv[])
{
    QGuiApplication *app = Aurora::Application::application(argc, argv);
    RuntimeDispatcher *dispatcher = RuntimeDispatcher::instance();

    dispatcher->onTaskStarted("Remind", [](const QString &backgroundTaskID) {
        remindAboutBeer();
    });

    dispatcher->onApplicationStarted([] {
        QQuickView *view = Aurora::Application::createView();
        view->setSource(Aurora::Application::pathTo("qml/dev.glazkov.BeerReminder.qml"));
        view->show();

        initRemindBackgroundTask();
    });

    return app->exec();
}
```

Разделение кода приложения и кода фоновой задачи

Указываем блок кода, который будет выполнен только в приложении

```
#include <RuntimeManager/RuntimeDispatcher>

int main(int argc, char *argv[])
{
    QGuiApplication *app = Aurora::Application::application(argc, argv);
    RuntimeDispatcher *dispatcher = RuntimeDispatcher::instance();

    dispatcher->onTaskStarted("Remind", [](const QString &backgroundTaskID) {
        remindAboutBeer();
    });

    dispatcher->onApplicationStarted([] {
        QQuickView *view = Aurora::Application::createView();
        view->setSource(Aurora::Application::pathTo("qml/dev.glazkov.BeerReminder.qml"));
        view->show();

        initRemindBackgroundTask();
    });

    return app->exec();
}
```



Инициализация фоновой задачи

Приложению необходимо единожды запустить фоновую задачу вручную

```
#include <RuntimeManager/RuntimeDispatcher>

int main(int argc, char *argv[])
{
    QGuiApplication *app = Aurora::Application::application(argc, argv);
    RuntimeDispatcher *dispatcher = RuntimeDispatcher::instance();

    dispatcher->onTaskStarted("Remind", [](const QString &backgroundTaskID) {
        remindAboutBeer();
    });

    dispatcher->onApplicationStarted([] {
        QQuickView *view = Aurora::Application::createView();
        view->setSource(Aurora::Application::pathTo("qml/dev.glazkov.BeerReminder.qml"));
        view->show();

        initRemindBackgroundTask();
    });

    return app->exec();
}
```

Инициализация фоновой задачи

Приложению необходимо единожды запустить фоновую задачу вручную

```
void initRemindBackgroundTask()
{
    Schedule schedule;
    schedule.onWeekDays({5});
    schedule.onHours({17});
    schedule.onMinutes({0});

    Task task("Remind");
    task.withSchedule(schedule);
    task.withAutostart(true);
    task.start([](const Error &error) {
        if (error) {
            qWarning() << "Error:" << error;
        }
    });
}
```

Инициализация фоновой задачи

Создаем планировщик для фоновой задачи с типом scheduled

```
void initRemindBackgroundTask()
{
    Schedule schedule;
    schedule.onWeekDays({5});
    schedule.onHours({17});
    schedule.onMinutes({0});

    Task task("Remind");
    task.withSchedule(schedule);
    task.withAutostart(true);
    task.start([](const Error &error) {
        if (error) {
            qWarning() << "Error:" << error;
        }
    });
}
```

Инициализация фоновой задачи

Указываем в какие дни недели запускать фоновую задачу

```
void initRemindBackgroundTask()
{
    Schedule schedule;
    schedule.onWeekDays({5});
    schedule.onHours({17});
    schedule.onMinutes({0});

    Task task("Remind");
    task.withSchedule(schedule);
    task.withAutostart(true);
    task.start([](const Error &error) {
        if (error) {
            qWarning() << "Error:" << error;
        }
    });
}
```

Инициализация фоновой задачи

Указываем в какие часы запускать фоновую задачу

```
void initRemindBackgroundTask()
{
    Schedule schedule;
    schedule.onWeekDays({5});
    schedule.onHours({17});
    schedule.onMinutes({0});

    Task task("Remind");
    task.withSchedule(schedule);
    task.withAutostart(true);
    task.start([](const Error &error) {
        if (error) {
            qWarning() << "Error:" << error;
        }
    });
}
```

Инициализация фоновой задачи

Указываем в какие минуты запускать фоновую задачу

```
void initRemindBackgroundTask()
{
    Schedule schedule;
    schedule.onWeekDays({5});
    schedule.onHours({17});
    schedule.onMinutes({0});

    Task task("Remind");
    task.withSchedule(schedule);
    task.withAutostart(true);
    task.start([](const Error &error) {
        if (error) {
            qWarning() << "Error:" << error;
        }
    });
}
```

Инициализация фоновой задачи

Создаем класс фоновой задачи, указывая ее название

```
void initRemindBackgroundTask()
{
    Schedule schedule;
    schedule.onWeekDays({5});
    schedule.onHours({17});
    schedule.onMinutes({0});

    Task task("Remind");
    task.withSchedule(schedule);
    task.withAutostart(true);
    task.start([](const Error &error) {
        if (error) {
            qWarning() << "Error:" << error;
        }
    });
}
```

Инициализация фоновой задачи

Привязываем фоновую задачу к планировщику

```
void initRemindBackgroundTask()
{
    Schedule schedule;
    schedule.onWeekDays({5});
    schedule.onHours({17});
    schedule.onMinutes({0});

    Task task("Remind");
    task.withSchedule(schedule);
    task.withAutostart(true);
    task.start([](const Error &error) {
        if (error) {
            qWarning() << "Error:" << error;
        }
    });
}
```

Инициализация фоновой задачи

Автоматическая инициализация фоновой задачи при перезагрузке устройства

```
void initRemindBackgroundTask()
{
    Schedule schedule;
    schedule.onWeekDays({5});
    schedule.onHours({17});
    schedule.onMinutes({0});

    Task task("Remind");
    task.withSchedule(schedule);
    task.withAutostart(true);
    task.start([](const Error &error) {
        if (error) {
            qWarning() << "Error:" << error;
        }
    });
}
```

Инициализация фоновой задачи

Запуск фоновой задачи

```
void initRemindBackgroundTask()
{
    Schedule schedule;
    schedule.onWeekDays({5});
    schedule.onHours({17});
    schedule.onMinutes({0});

    Task task("Remind");
    task.withSchedule(schedule);
    task.withAutostart(true);
    task.start([](const Error &error) {
        if (error) {
            qWarning() << "Error:" << error;
        }
    });
}
```

Инициализация фоновой задачи

Запуск фоновой задачи

```
void initRemindBackgroundTask()
{
    Schedule schedule;
    schedule.onWeekDays({5});
    schedule.onHours({17});
    schedule.onMinutes({0});

    Task task("Remind");
    task.withSchedule(schedule);
    task.withAutostart(true);
    task.start([](const Error &error) {
        if (error) {
            qWarning() << "Error:" << error;
        }
    });
}
```

Реализация уведомления через фоновую задачу

```
#include <RuntimeManager/RuntimeDispatcher>

int main(int argc, char *argv[])
{
    QGuiApplication *app = Aurora::Application::application(argc, argv);
    RuntimeDispatcher *dispatcher = RuntimeDispatcher::instance();

    dispatcher->onTaskStarted("Remind", [](const QString &backgroundTaskID) {
        remindAboutBeer();
    });

    dispatcher->onApplicationStarted([] {
        QQuickView *view = Aurora::Application::createView();
        view->setSource(Aurora::Application::pathTo("qml/dev.glazkov.BeerReminder.qml"));
        view->show();

        initRemindBackgroundTask();
    });

    return app->exec();
}
```

Реализация уведомления через фоновую задачу

Конфигурация уведомления

```
void remindAboutBeer()
{
    notification.setArguments({
        "Пивная напоминалка",
        quint32(0),
        Aurora::Application::pathTo("qml/images/icon.png").toString(),
        "Пятница",
        "Пришло время позвать друзей пить пиво",
        QStringList{
            "findBar", "Найти бар",
            "callFriend", "Позвонить лучшему другу",
            "default", "On notification clicked",
        },
        QVariantMap{
            {"x-nemo-remote-action-findBar", "https://friday.bars.ru"},
            {"x-nemo-remote-action-callFriend", "tel:89998887766"},
            {"x-nemo-remote-action-default", "aurora-dev.glazkov.BeerReminder:BeerPage"},
        },
        qint32(10 * 1000),
    });
}
```

Реализация уведомления через фоновую задачу

Указываем имя приложения

```
void remindAboutBeer()
{
    notification.setArguments({
        "Пивная напоминалка"
        quint32(0),
        Aurora::Application::pathTo("qml/images/icon.png").toString(),
        "Пятница",
        "Пришло время позвать друзей пить пиво",
        QStringList{
            "findBar", "Найти бар",
            "callFriend", "Позвонить лучшему другу",
            "default", "On notification clicked",
        },
        QVariantMap{
            {"x-nemo-remote-action-findBar", "https://friday.bars.ru"},
            {"x-nemo-remote-action-callFriend", "tel:89998887766"},
            {"x-nemo-remote-action-default", "aurora-dev.glazkov.BeerReminder:BeerPage"},
        },
        qint32(10 * 1000),
    });
}
```

Реализация уведомления через фоновую задачу

Указываем иконку уведомления

```
void remindAboutBeer()
{
    notification.setArguments({
        "Пивная напоминалка",
        quint32(0),
        Aurora::Application::pathTo("qml/images/icon.png").toString(),
        "Пятница",
        "Пришло время позвать друзей пить пиво",
        QStringList{
            "findBar", "Найти бар",
            "callFriend", "Позвонить лучшему другу",
            "default", "On notification clicked",
        },
        QVariantMap{
            {"x-nemo-remote-action-findBar", "https://friday.bars.ru"},
            {"x-nemo-remote-action-callFriend", "tel:89998887766"},
            {"x-nemo-remote-action-default", "aurora-dev.glazkov.BeerReminder:BeerPage"},
        },
        qint32(10 * 1000),
    });
}
```

Реализация уведомления через фоновую задачу

Указываем заголовок уведомления

```
void remindAboutBeer()
{
    notification.setArguments({
        "Пивная напоминалка",
        quint32(0),
        Aurora::Application::pathTo("qml/images/icon.png").toString(),
        "Пятница",
        "Пришло время позвать друзей пить пиво",
        QStringList{
            "findBar", "Найти бар",
            "callFriend", "Позвонить лучшему другу",
            "default", "On notification clicked",
        },
        QVariantMap{
            {"x-nemo-remote-action-findBar", "https://friday.bars.ru"},
            {"x-nemo-remote-action-callFriend", "tel:89998887766"},
            {"x-nemo-remote-action-default", "aurora-dev.glazkov.BeerReminder:BeerPage"},
        },
        qint32(10 * 1000),
    });
}
```

Реализация уведомления через фоновую задачу

Указываем текст уведомления

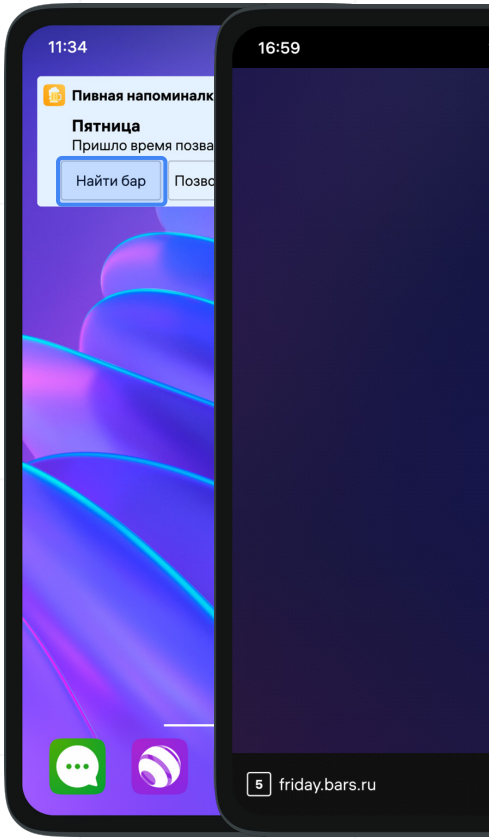
```
void remindAboutBeer()
{
    notification.setArguments({
        "Пивная напоминалка",
        quint32(0),
        Aurora::Application::pathTo("qml/images/icon.png").toString(),
        "Пятница",
        "Пришло время позвать друзей пить пиво"
        QStringList{
            "findBar", "Найти бар",
            "callFriend", "Позвонить лучшему другу",
            "default", "On notification clicked",
        },
        QVariantMap{
            {"x-nemo-remote-action-findBar", "https://friday.bars.ru"},
            {"x-nemo-remote-action-callFriend", "tel:89998887766"},
            {"x-nemo-remote-action-default", "aurora-dev.glazkov.BeerReminder:BeerPage"},
        },
        qint32(10 * 1000),
    });
}
```

Реализация уведомления через фоновую задачу

Открытие браузера по кнопке через интент OpenURI

```
void remindAboutBeer()
{
    notification.setArguments({
        "Пивная напоминалка",
        quint32(0),
        Aurora::Application::pathTo("qml/images/icon.png").toString(),
        "Пятница",
        "Пришло время позвать друзей пить пиво",
        QStringList{
            "findBar", "Найти бар",
            "callFriend", "Позвонить лучшему другу",
            "default", "On notification clicked",
        },
        QVariantMap{
            {"x-nemo-remote-action-findBar", "https://friday.bars.ru"},
            {"x-nemo-remote-action-callFriend", "tel:89998887766"},
            {"x-nemo-remote-action-default", "aurora-dev.glazkov.BeerReminder:BeerPage"},
        },
        qint32(10 * 1000),
    });
}
```

Вызов интента OpenURI через Runtime Manager

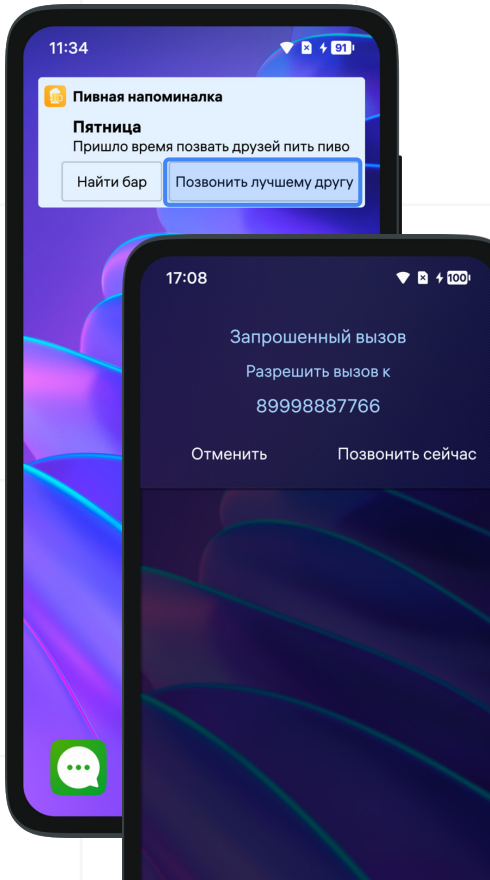


Реализация уведомления через фоновую задачу

Открытие телефона по кнопке через интент OpenURI

```
void remindAboutBeer()
{
    notification.setArguments({
        "Пивная напоминалка",
        quint32(0),
        Aurora::Application::pathTo("qml/images/icon.png").toString(),
        "Пятница",
        "Пришло время позвать друзей пить пиво",
        QStringList{
            "findBar", "Найти бар",
            "callFriend", "Позвонить лучшему другу",
            "default", "On notification clicked",
        },
        QVariantMap{
            {"x-nemo-remote-action-findBar", "https://friday.bars.ru"},
            {"x-nemo-remote-action-callFriend", "tel:89998887766"},
            {"x-nemo-remote-action-default", "aurora-dev.glazkov.BeerReminder:BeerPage"},
        },
        qint32(10 * 1000),
    });
}
```

Вызов интента OpenURI через Runtime Manager



Реализация уведомления через фоновую задачу

Открытие страницы приложения через интент OpenURI

```
void remindAboutBeer()
{
    notification.setArguments({
        "Пивная напоминалка",
        quint32(0),
        Aurora::Application::pathTo("qml/images/icon.png").toString(),
        "Пятница",
        "Пришло время позвать друзей пить пиво",
        QStringList{
            "findBar", "Найти бар",
            "callFriend", "Позвонить лучшему другу",
            "default", "On notification clicked"
        },
        QVariantMap{
            {"x-nemo-remote-action-findBar", "https://friday.bars.ru"},
            {"x-nemo-remote-action-callFriend", "tel:89998887766"},
            {"x-nemo-remote-action-default", "aurora-dev.glazkov.BeerReminder:BeerPage"}
        },
        qint32(10 * 1000),
    });
}
```

Вызов интента OpenURI через Runtime Manager

Desktop файл приложения для ОС Аврора

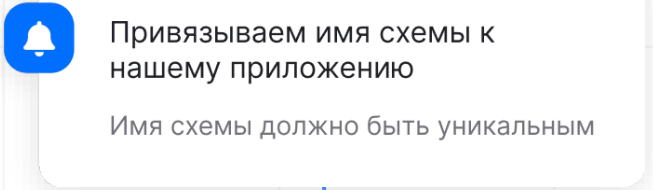
Добавление поддержки кастомной схемы для интента OpenURI

```
[Desktop Entry]
Type=Application
Exec=/usr/bin/dev.glazkov.BeerReminder
Icon=dev.glazkov.BeerReminder
Name=Friday Beer
MimeType=x-scheme-handler/aurora-dev.glazkov.beerreminder

[X-Application]
OrganizationName=dev.glazkov
ApplicationName=BeerReminder
Intents=OpenURI
```

Desktop файл приложения для ОС Аврора

Добавление поддержки кастомной схемы для интента OpenURI

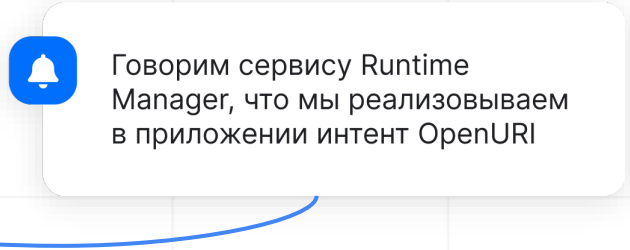
```
[Desktop Entry]
Type=Application
Exec=/usr/bin/dev.glazkov.BeerReminder
Icon=dev.glazkov.BeerReminder
Name=Friday Beer
MimeType=x-scheme-handler/aurora-dev.glazkov.beerreminder

[X-Application]
OrganizationName=dev.glazkov
ApplicationName=BeerReminder
Intents=OpenURI
```



Desktop файл приложения для ОС Аврора

Добавление поддержки кастомной схемы для интента OpenURI

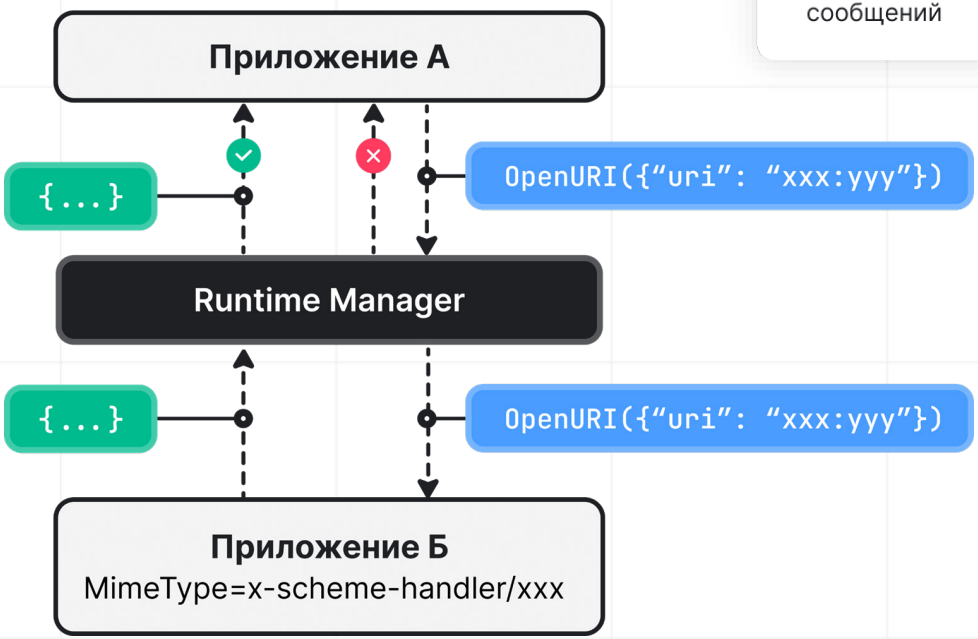
```
[Desktop Entry]
Type=Application
Exec=/usr/bin/dev.glazkov.BeerReminder
Icon=dev.glazkov.BeerReminder
Name=Friday Beer
MimeType=x-scheme-handler/aurora-dev.glazkov.beerreminder

[X-Application]
OrganizationName=dev.glazkov
ApplicationName=BeerReminder
Intents=OpenURI
```



Механизм интентов в ОС Аврора

Взаимодействие двух приложений

1. Взаимодействие через интенты происходит при помощи JSON сообщений



Реализация обработчика интента на языке QML

MainPage.qml

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentHandler {
        intentName: "OpenURI"
        onInvoked: {
            var page = data["uri"].split(':')[1]
            pageStack.replace(Qt.resolvedUrl(page + ".qml"))

            response.send({})
        }
    }
}
```

Реализация обработчика интента на языке QML

Подключаем библиотеку Runtime Manager

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentHandler {
        intentName: "OpenURI"
        onInvoked: {
            var page = data["uri"].split(':')[1]
            pageStack.replace(Qt.resolvedUrl(page + ".qml"))

            response.send({})
        }
    }
}
```

Реализация обработчика интента на языке QML

Добавляем объект IntentHandler

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentHandler {
        intentName: "OpenURI"
        onInvoked: {
            var page = data["uri"].split(':')[1]
            pageStack.replace(Qt.resolvedUrl(page + ".qml"))

            response.send({})
        }
    }
}
```

Реализация обработчика интента на языке QML

Указываем имя обрабатываемого интента

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentHandler {
        intentName: "OpenURI"
        onInvoked: {
            var page = data["uri"].split(':')[1]
            pageStack.replace(Qt.resolvedUrl(page + ".qml"))

            response.send({})
        }
    }
}
```

Реализация обработчика интента на языке QML

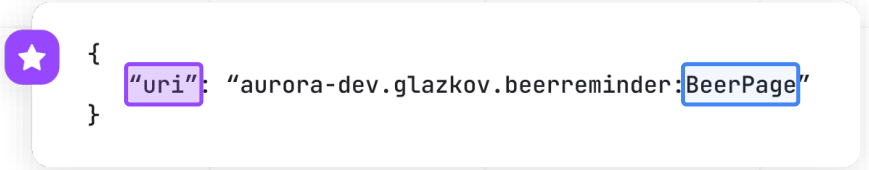
Указываем реализацию интента на языке JavaScript

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentHandler {
        intentName: "OpenURI"
        onInvoked: {
            var page = data["uri"].split(':')[1]
            pageStack.replace(Qt.resolvedUrl(page + ".qml"))

            response.send({})
        }
    }
}
```

Реализация обработчика интента на языке QML

Получаем название страницы из входных данных

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentHandler {
        intentName: "OpenURI"
        onInvoked: {
            var page = data["uri"].split(':')[1]
            pageStack.replace(Qt.resolvedUrl(page + ".qml"))

            response.send({})
        }
    }
}
```



Реализация обработчика интента на языке QML

Открываем необходимую страницу

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentHandler {
        intentName: "OpenURI"
        onInvoked: {
            var page = data["uri"].split(':')[1]
            pageStack.replace(Qt.resolvedUrl(page + ".qml"))

            response.send({})
        }
    }
}
```

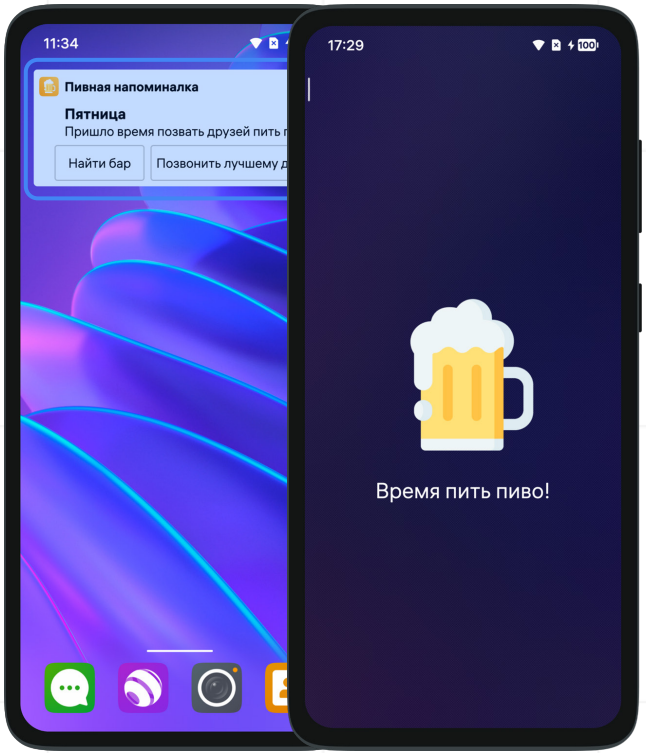
Реализация обработчика интента на языке QML

Отправляем ответ в формате JSON вызывающей стороне

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentHandler {
        intentName: "OpenURI"
        onInvoked: {
            var page = data["uri"].split(':')[1]
            pageStack.replace(Qt.resolvedUrl(page + ".qml"))

            response.send({})
        }
    }
}
```

Реализация обработчика интента на языке QML

Итоговый результат



Реализация вызова интента на языке QML

Подключаем библиотеку Runtime Manager

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentsInvoker {
        id: invoker
        onReplyReceived: {
            if (error) { console.log(error) }
            else       { console.log(reply) }
        }
    }

    Button {
        text: "Я устал ждать"
        onClicked: {
            invoker.invoke("OpenURI", {}, {
                "uri": "aurora-dev.glazkov.beerreminder:BeerPage"
            })
        }
    }
}
```

Реализация вызова интента на языке QML

Создаем объект IntentsInvoker

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentsInvoker {
        id: invoker
        onReplyReceived: {
            if (error) { console.log(error) }
            else       { console.log(reply) }
        }
    }

    Button {
        text: "Я устал ждать"
        onClicked: {
            invoker.invoke("OpenURI", {}, {
                "uri": "aurora-dev.glazkov.beerreminder:BeerPage"
            })
        }
    }
}
```

Реализация вызова интента на языке QML

Реализовываем обработчик ответных сообщений

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentsInvoker {
        id: invoker
        onReplyReceived: {
            if (error) { console.log(error) }
            else       { console.log(reply) }
        }
    }

    Button {
        text: "Я устал ждать"
        onClicked: {
            invoker.invoke("OpenURI", {}, {
                "uri": "aurora-dev.glazkov.beerreminder:BeerPage"
            })
        }
    }
}
```

Реализация вызова интента на языке QML

Обработка ошибки вызова интента

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentsInvoker {
        id: invoker
        onReplyReceived: {
            if (error) { console.log(error) }
            else       { console.log(reply) }
        }
    }

    Button {
        text: "Я устал ждать"
        onClicked: {
            invoker.invoke("OpenURI", {}, {
                "uri": "aurora-dev.glazkov.beerreminder:BeerPage"
            })
        }
    }
}
```

Реализация вызова интента на языке QML

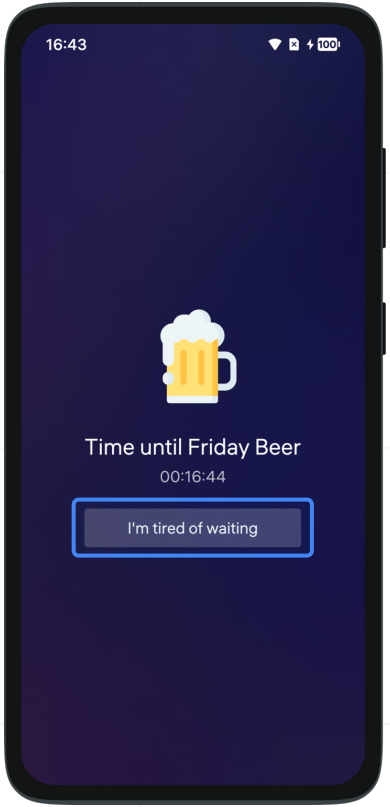
Обработка ответа в формате JSON

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentsInvoker {
        id: invoker
        onReplyReceived: {
            if (error) { console.log(error) }
            else       { console.log(reply) }
        }
    }

    Button {
        text: "Я устал ждать"
        onClicked: {
            invoker.invoke("OpenURI", {}, {
                "uri": "aurora-dev.glazkov.beerreminder:BeerPage"
            })
        }
    }
}
```

Реализация вызова интента на языке QML

Добавляем кнопку для вызова интента

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentsInvoker {
        id: invoker
        onReplyReceived: {
            if (error) { console.log(error) }
            else       { console.log(reply) }
        }
    }

    Button {
        text: "Я устал ждать"
        onClicked: {
            invoker.invoke("OpenURI", {}, {
                "uri": "aurora-dev.glazkov.beerreminder:BeerPage"
            })
        }
    }
}
```



Реализация вызова интента на языке QML

Указываем имя интента

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentsInvoker {
        id: invoker
        onReplyReceived: {
            if (error) { console.log(error) }
            else       { console.log(reply) }
        }
    }

    Button {
        text: "Я устал ждать"
        onClicked: {
            invoker.invoke("OpenURI", {}, {
                "uri": "aurora-dev.glazkov.beerreminder:BeerPage"
            })
        }
    }
}
```

Документация
по интентам

Реализация вызова интента на языке QML

Указываем дополнительные параметры в JSON формате

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentsInvoker {
        id: invoker
        onReplyReceived: {
            if (error) { console.log(error) }
            else       { console.log(reply) }
        }
    }

    Button {
        text: "Я устал ждать"
        onClicked: {
            invoker.invoke("OpenURI", {}, {
                "uri": "aurora-dev.glazkov.beerreminder:BeerPage"
            })
        }
    }
}
```

Документация по интентам

Реализация вызова интента на языке QML

Указываем данные для интента в JSON формате

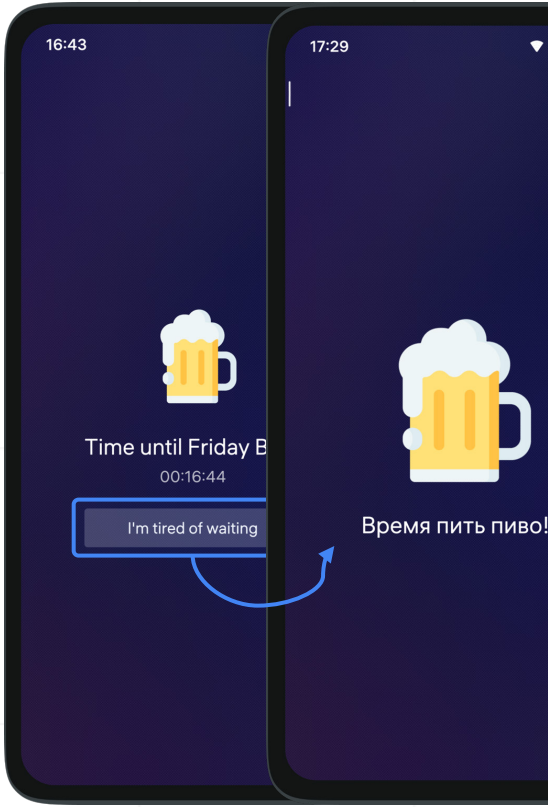
```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentsInvoker {
        id: invoker
        onReplyReceived: {
            if (error) { console.log(error) }
            else       { console.log(reply) }
        }
    }

    Button {
        text: "Я устал ждать"
        onClicked: {
            invoker.invoke("OpenURI", {}, {
                "uri": "aurora-dev.glazkov.beerreminder:BeerPage"
            })
        }
    }
}
```

Документация
по интентам

Реализация вызова интента на языке QML

Вызываем интент при клике на кнопку

```
import ru.auroraos.RuntimeManager 1.0

Page {
    IntentsInvoker {
        id: invoker
        onReplyReceived: {
            if (error) { console.log(error) }
            else       { console.log(reply) }
        }
    }

    Button {
        text: "Я устал ждать"
        onClicked: {
            invoker.invoke("OpenURI", {}, {
                "uri": "aurora-dev.glazkov.beerreminder:BeerPage"
            })
        }
    }
}
```



Пакетный менеджер APM

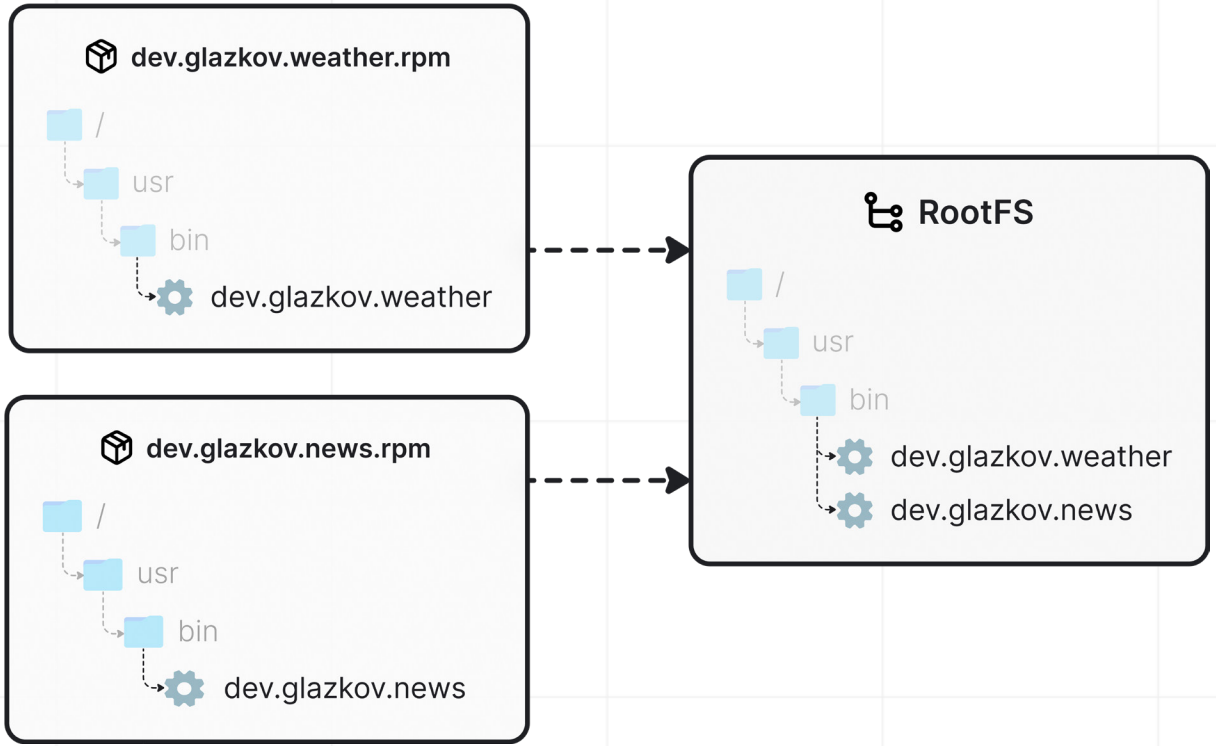
Новый пакетный менеджер для установки сторонних приложений

Пакетный менеджер APM

Гибкая настройка правил установки приложений

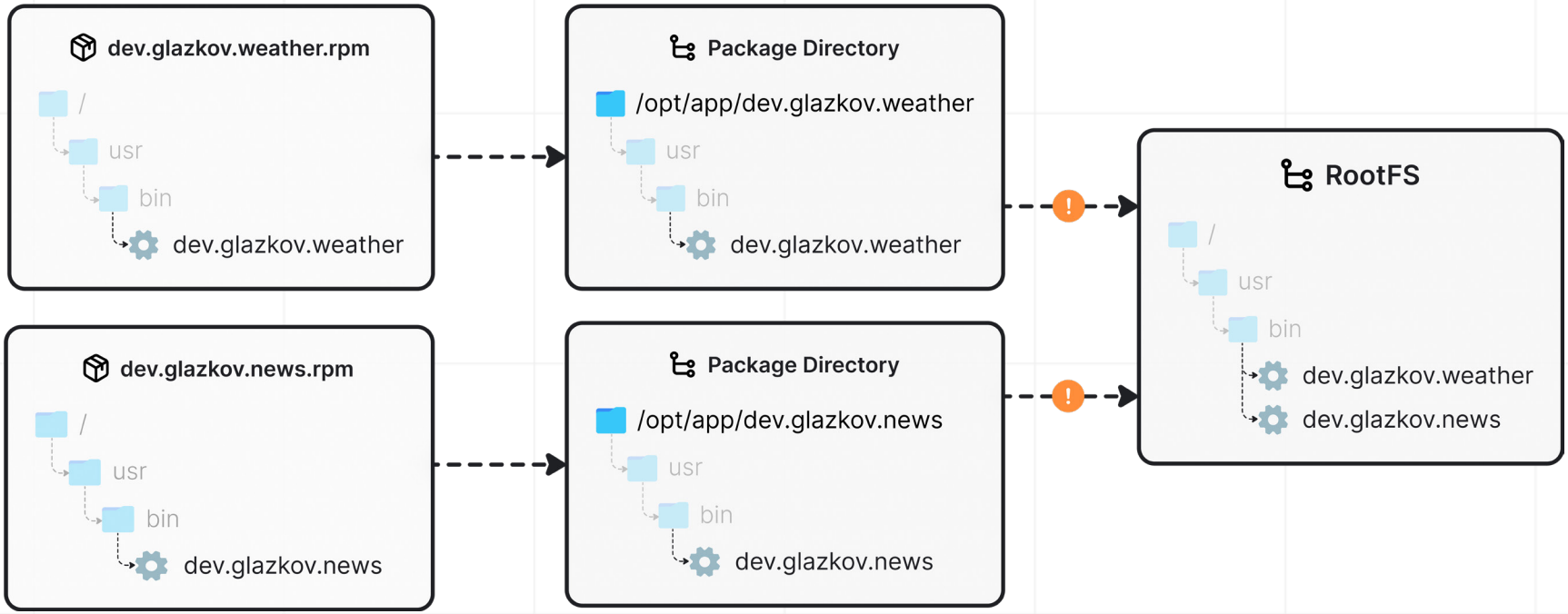
Процесс установки приложений через RPM

RPM напрямую модифицирует корневую файловую систему



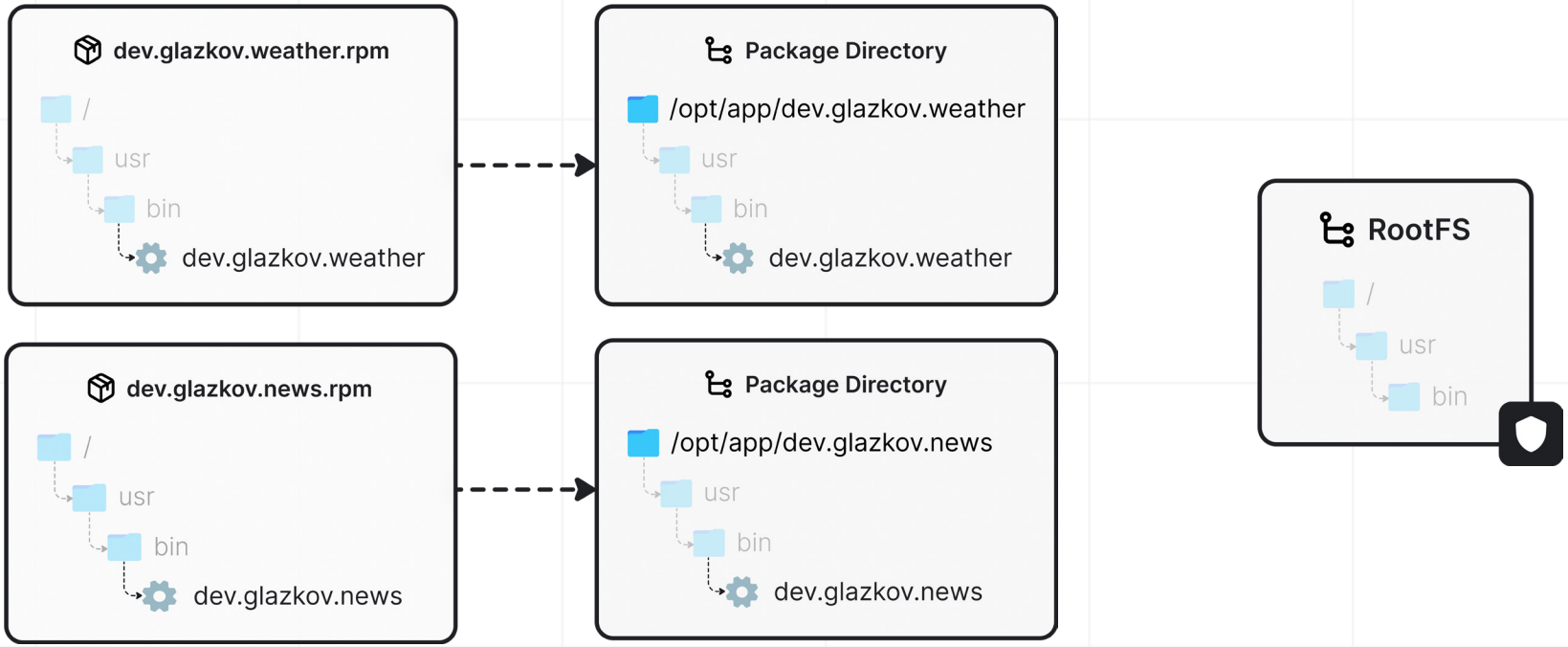
Процесс установки приложений через APM

APM устанавливает файлы в раздельные директории



Процесс установки приложений через APM

Позволит в будущем внедрить новые механизмы безопасности



Пакетный менеджер APM

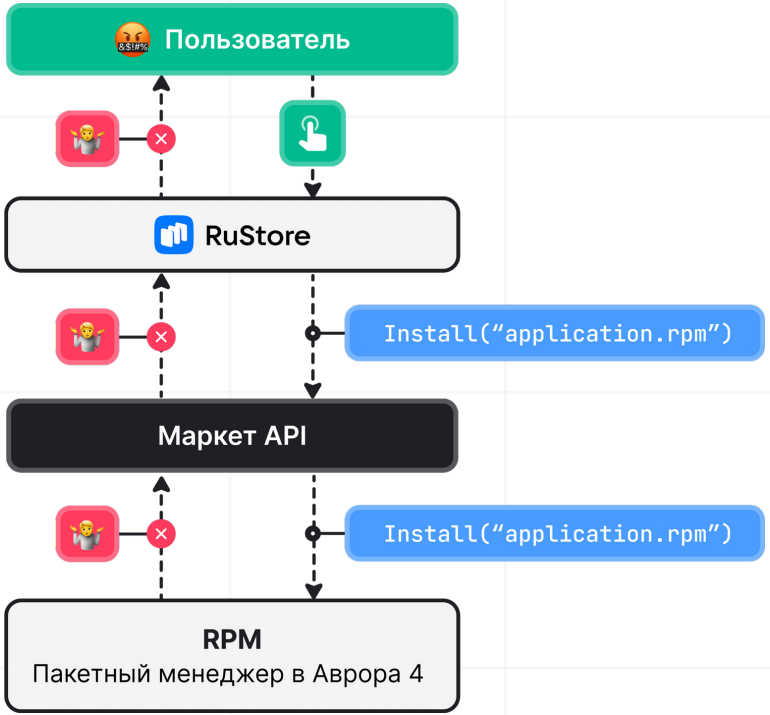
Бекенд для Маркет API

Появление глобального магазина приложений

Появилась необходимость в понятном и удобном API для маркетов

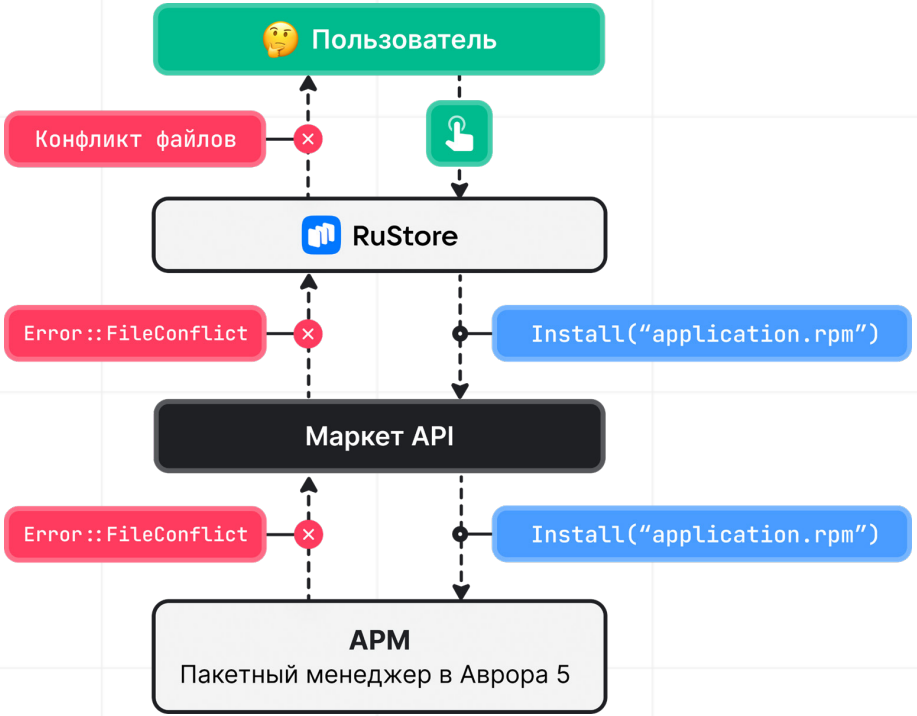
Пользовательский опыт

Если бы Маркет API был на базе RPM



Пользовательский опыт

В Аврора 5 с пакетным менеджером APM



Концепция доверенных источников

Новый механизм безопасности

Аврора 4.0

Жизнь до глобальных магазинов приложений

У клиента есть желание организовать закрытый контур

Мы выпускаем для клиента сертификат, которым клиент подписывает доверенные им приложения

Клиент организует непубличный магазин с доверенными приложениями

Концепция доверенных источников

Зачем что-то менять, если всех всё устраивало?

Аврора 5.0

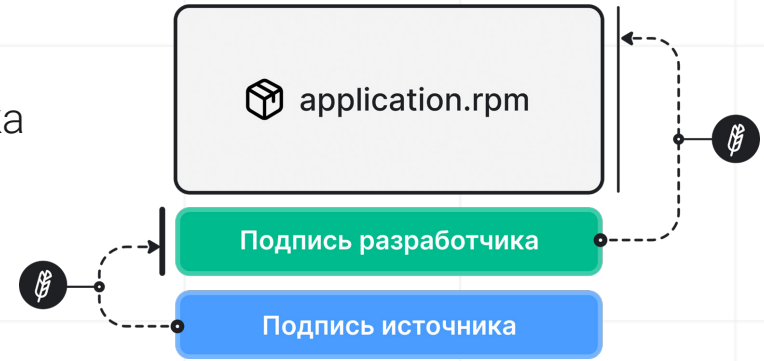
Наши цели, которые были поставлены при проектировании механизма доверенных источников

Сохранить и усилить возможности контроля замкнутости среды

Устранить недоработки механизма клиентской подписи

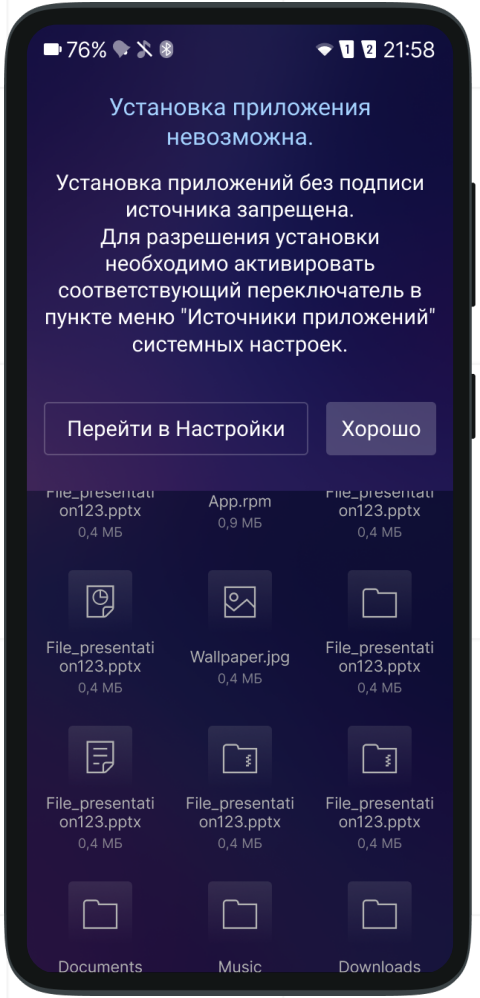
Быть готовыми к глобальным магазинам приложений

1. Клиентская подпись → Подпись источника
2. Функциональное предназначение второй подписи было сохранено



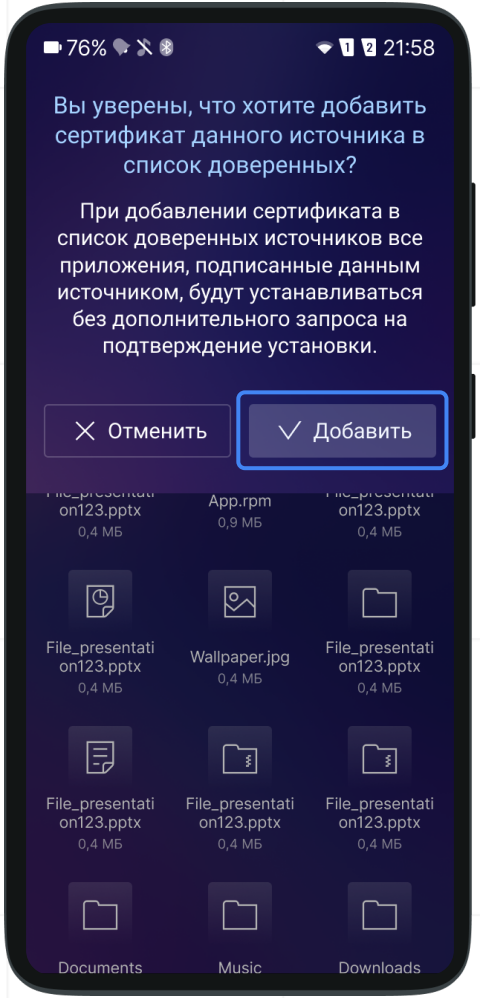
Подпись источника является **обязательной!**

За исключением, если администратор **явно разрешил** установку пакетов без подписи источника в настройках



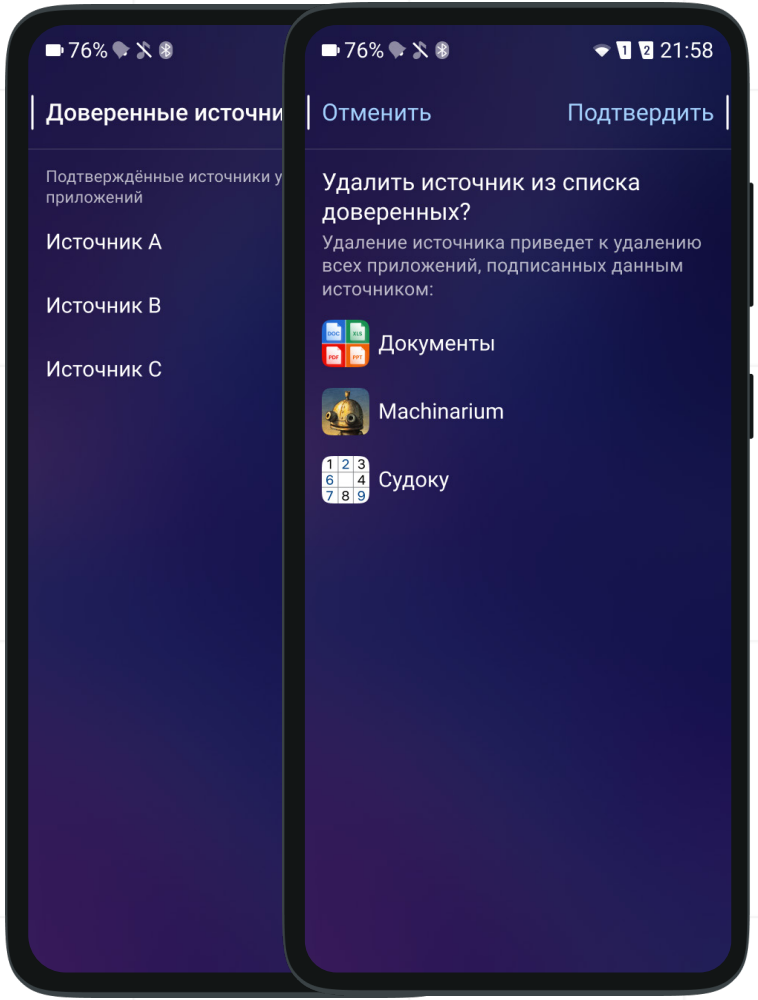
На устройстве могут находиться приложения **только из доверенных источников!**

Пока администратор не добавил источник приложения в список доверенных - установка приложения **невозможна**

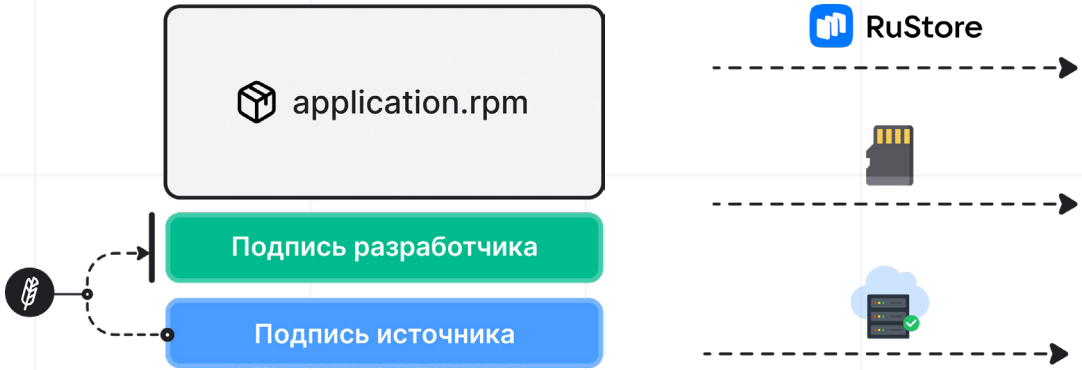


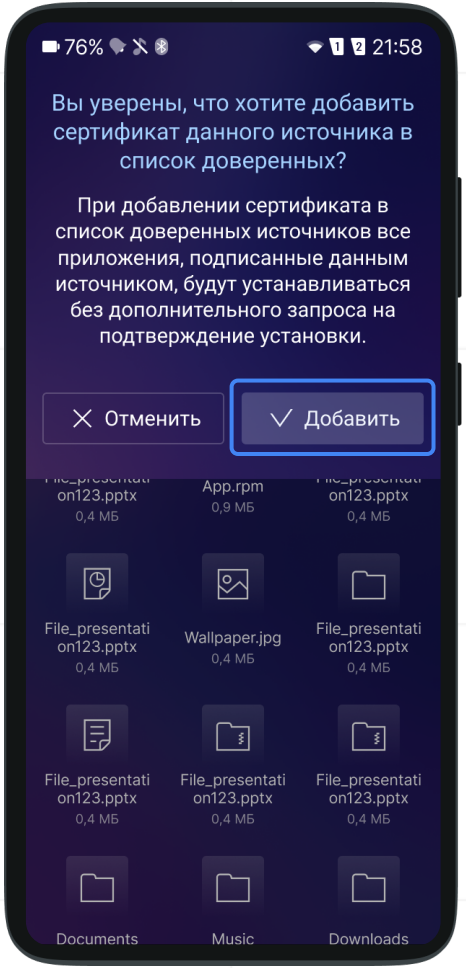
Список приложений на устройстве формируется **гибким перечнем источников**

При удалении источника из списка доверенных **удаляются все приложения**, привязанные к данному источнику



Доверие к источнику
приложения, а не к каналу поставки





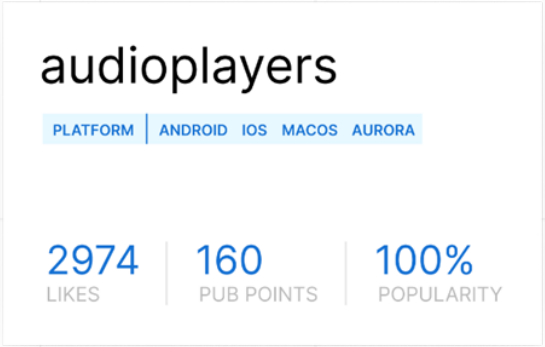
RuStore

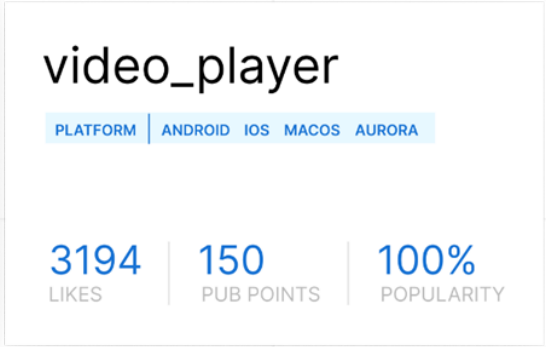
Является доверенным источником приложений в ОС Аврора

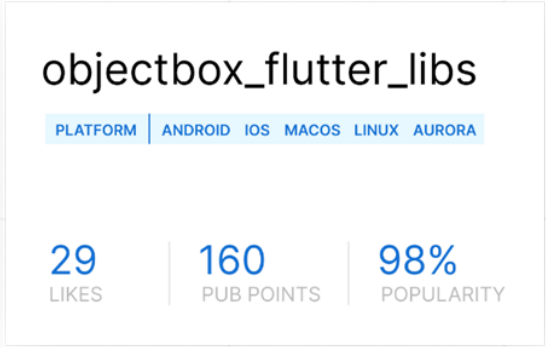
Прогресс в поддержке Flutter

Поддержка фреймворка Flutter

Добавлена поддержка новых плагинов







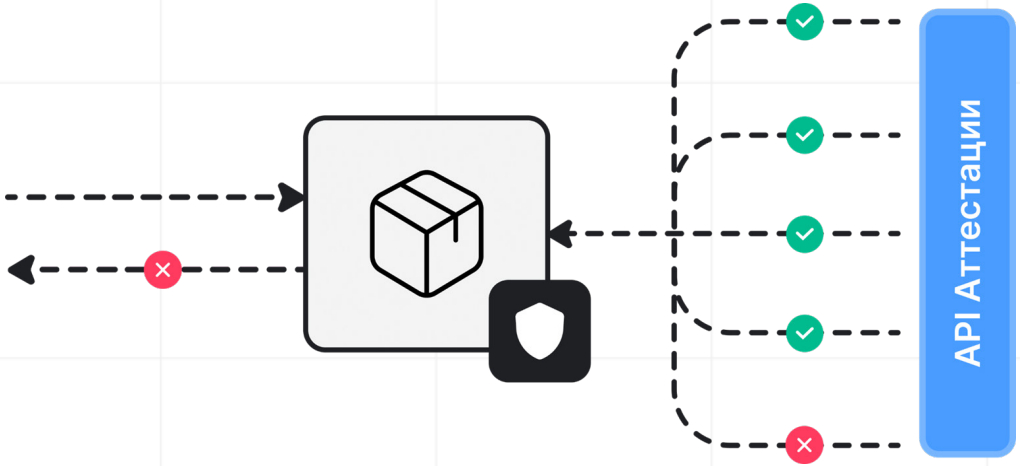
Поддержка фреймворка Flutter

- Flutter 3.24.0
- Поддержка эмулятора ОС Аврора
- Поддержка Hot Reload 🔥
- Поддержка Flutter Devices
- Поддержка расширения VSCode
- Поддержка SafeArea
- Исправление багов

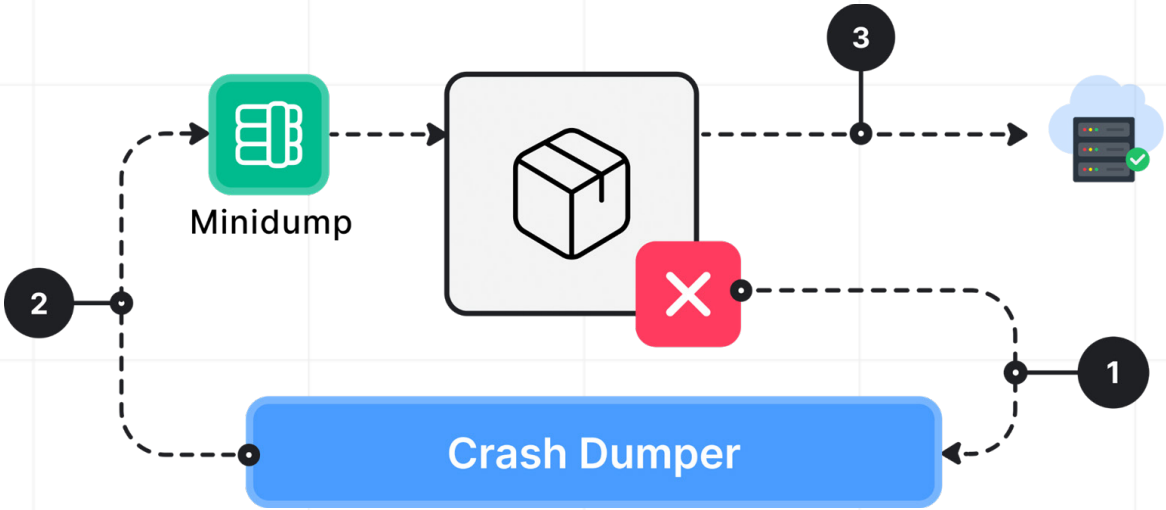
Flutter для ОС Аврора

Цели и планы по развитию ОС Аврора

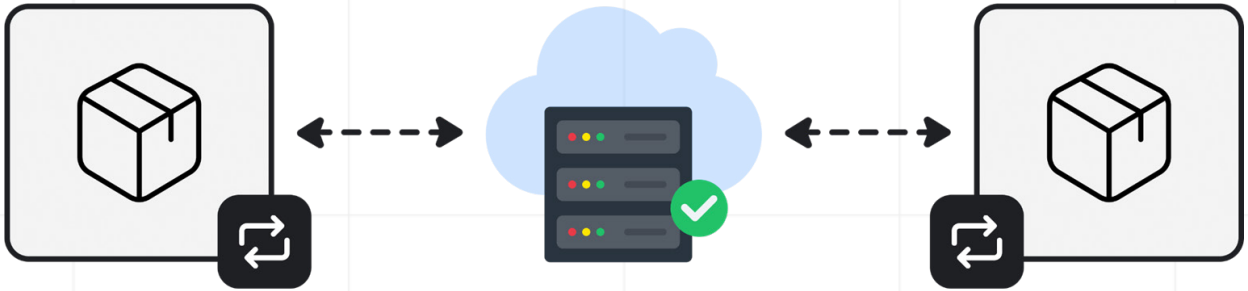
API Аттестации



API получения crash-литики

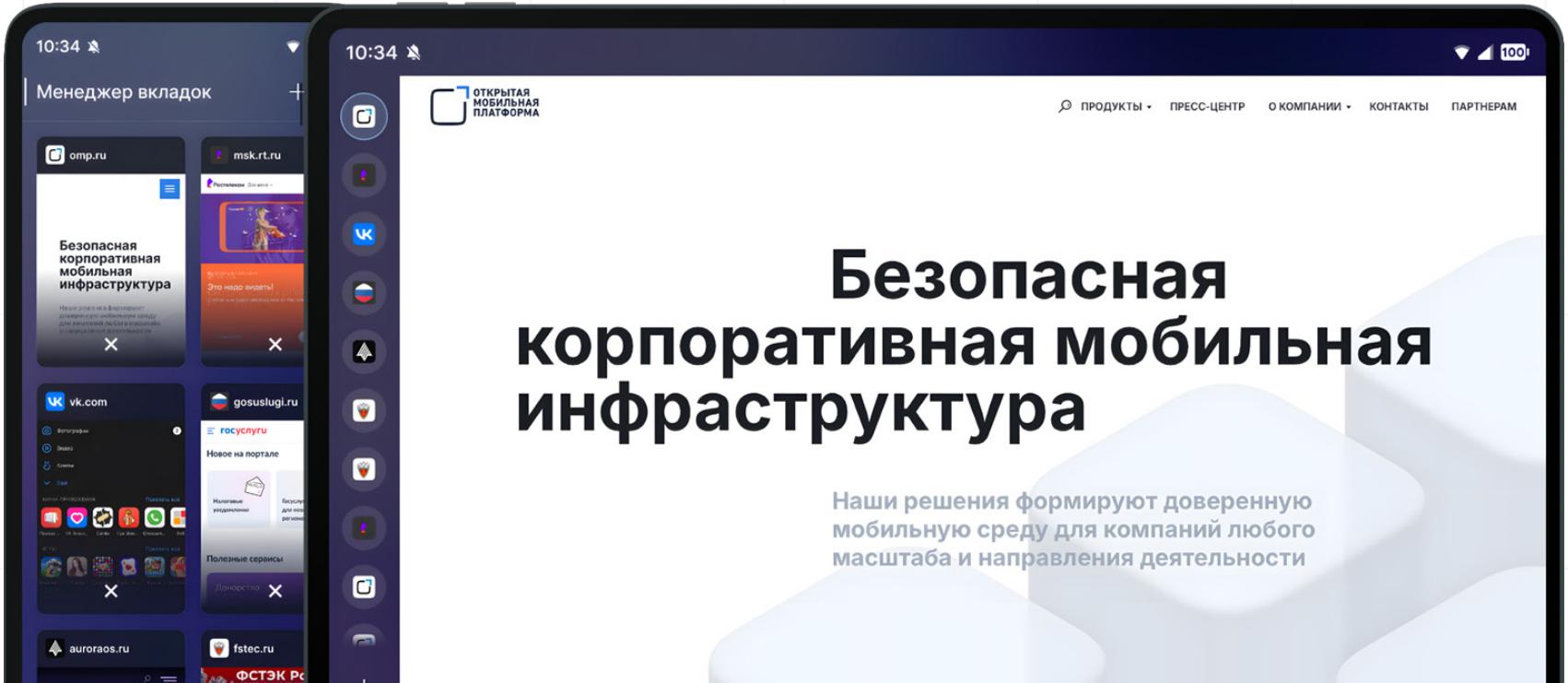


Удаленная конфигурация приложений

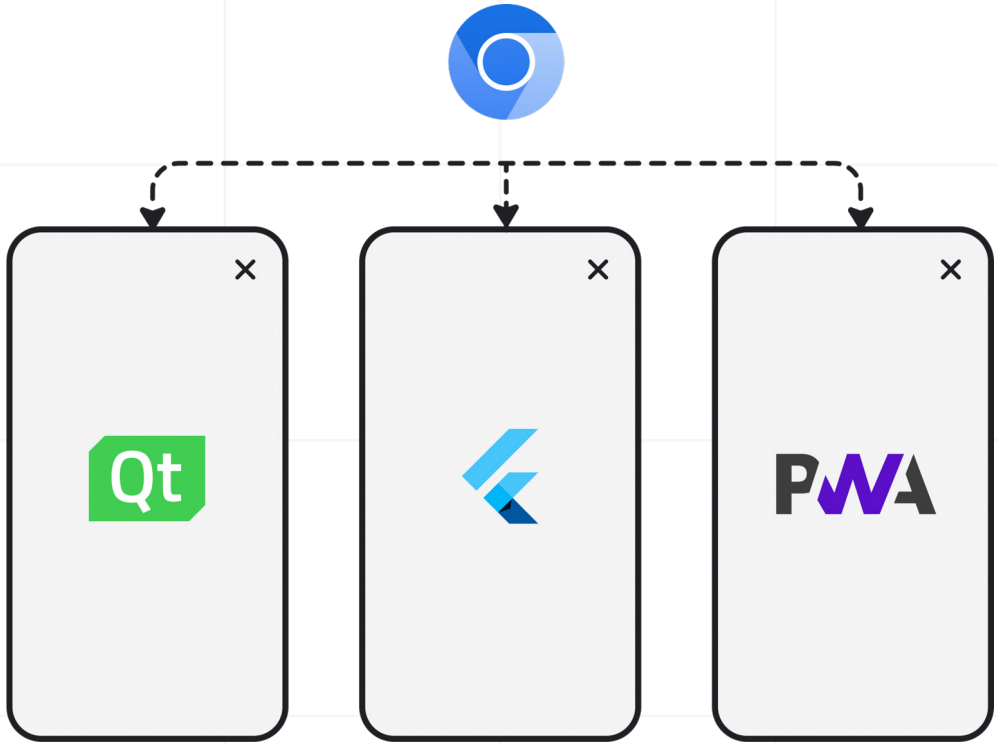


Браузер на базе Chromium

В ближайших версиях ОС Аврора 5

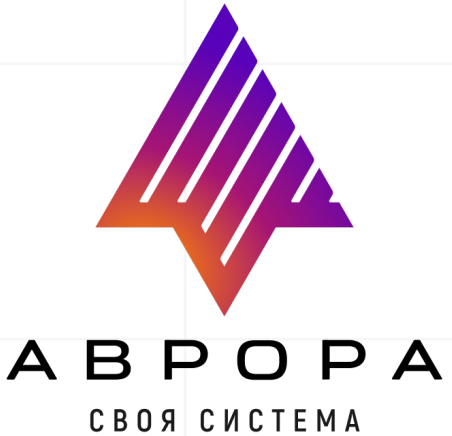


Webview и PWA на базе Chromium



Заключение

- Аврора - современная мобильная операционная система и сервисы
- Команда «Открытой мобильной платформы» активно развивает средства разработки и функционал ОС Аврора
- Под ОС Аврора активно разрабатываются новые приложения



Полезные ссылки

Документация
Runtime Manager API

Документация
Flutter для ОС Аврора

Телеграм канал для
разработчиков

https://github.com/dietime

Спасибо за внимание

ОТКРЫТАЯ МОБИЛЬНАЯ ПЛАТФОРМА

Глазков Денис

Старший инженер-разработчик отдела разработки ОС

github.com/dietime youtube.com/@glazzkoff dev-support@omp.ru