Пишем нативно под Аврора ОС как под iOS: Clean-архитектура, Coordinator и UDF

- Супрун Денис Алексеевич, 36 лет, индивидуальный предприниматель
- окончил физический факультет РГУ (ЮФУ), степень магистра по специальности Цифровая Обработка Сигналов
- стаж разработки под Android и iOS более 10 лет
- последние 5 лет mobile tech lead в Liloo Labs Inc.

Дорожная карта

1. Аврора ОС
2. Инструменты разработчика
3. CLEAN архитектура и Coordinator
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. DI
8. Итоги

Дорожная карта

1. Аврора ОС
2. Инструменты разработчика
3. CLEAN архитектура и Coordinator
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. DI
8. Итоги

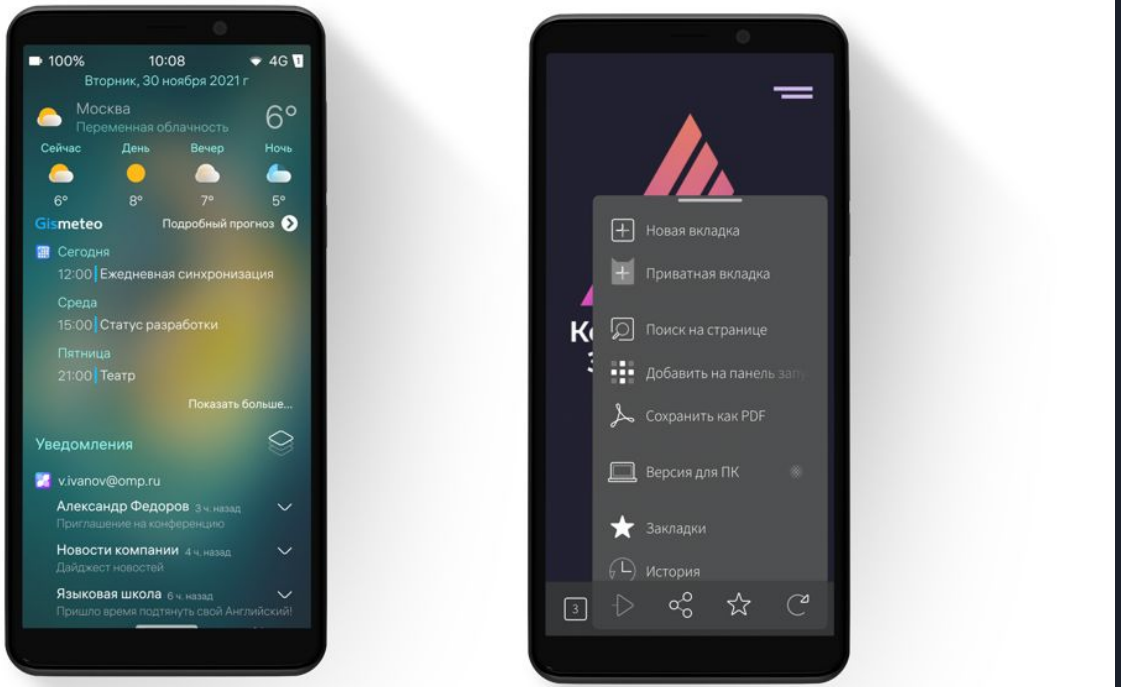
Аврора ОС

- Не AOSP
- Разрабатывается компанией Открытая Мобильная Платформа
- Корпоративная. Защищённая. Своя

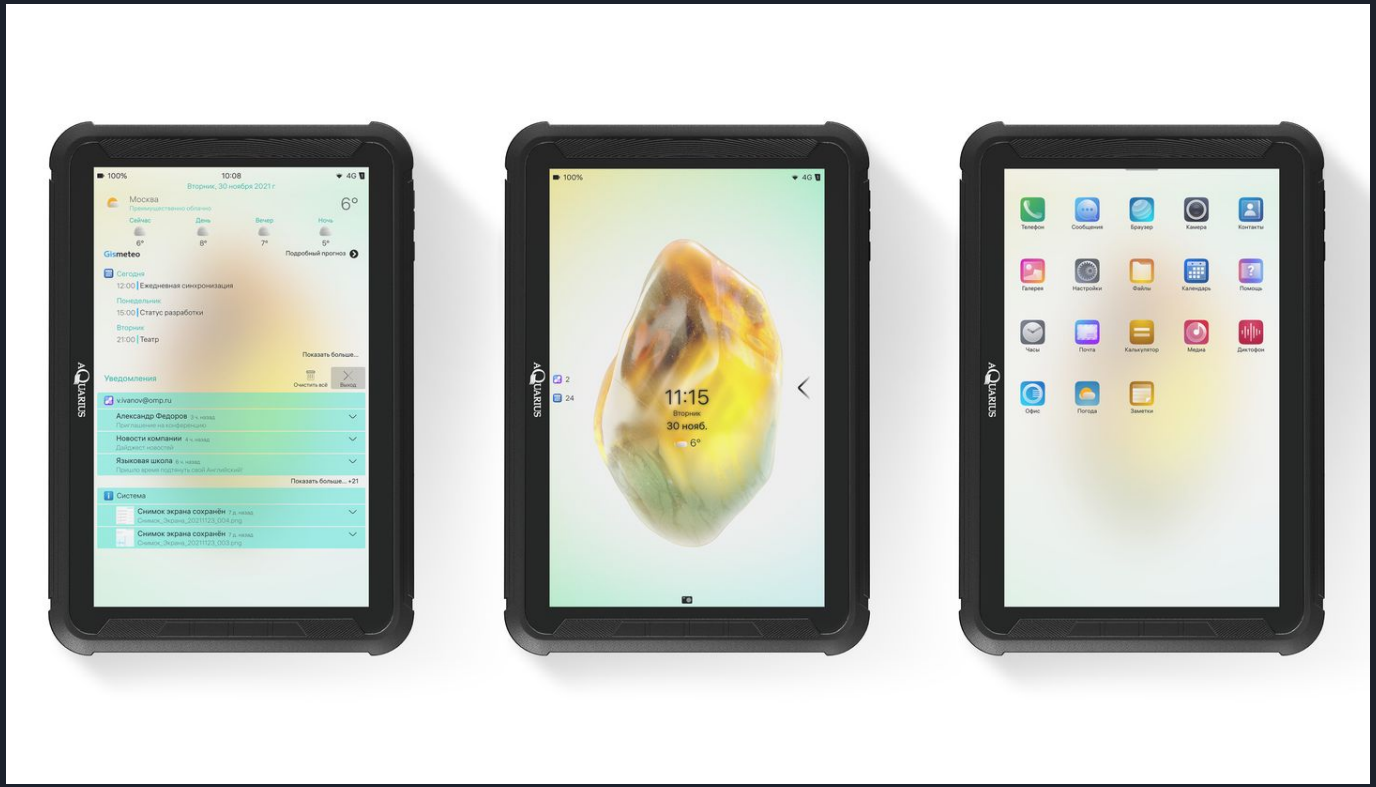
Аврора ОС


Телефон
Сообщения
Браузер
Камера
Контакты
Галерея
Настройки
Помощь
Файлы
Календарь
Почта
Медиа
Диктофон
Офис
Часы
Калькулятор
Погода
Заметки
Terminal
Просмотр PDF
100%
4G
11:15
Вторник
30 нояб.
6°
100%
10:08
4G
6°
Москва
3
Непрочит. письма
0:00
Картинки с выставки

Аврора ОС


100%
10:08
4G
Вторник, 30 ноября 2021 г
Москва
Переменная облачность
6°
Сейчас
День
Вечер
Ночь
6°
8°
7°
5°
Gismeteo
Подробный прогноз
Сегодня
12:00 Ежедневная синхронизация
Среда
15:00 Статус разработки
Пятница
21:00 Театр
Показать больше...
Уведомления
v.ivanov@omp.ru
Александр Федоров 3 ч. назад
Приглашение на конференцию
Новости компании 4 ч. назад
Дайджест новостей
Языковая школа 6 ч. назад
Пришло время подтянуть свой Английский!
Новая вкладка
Приватная вкладка
Поиск на странице
Сохранить как PDF
Версия для ПК
Закладки
История

Аврора ОС


100%
10:08
4G
Вторник, 30 ноября 2021 г
Москва
6°
Сейчас
День
Вечер
Ночь
6°
8°
7°
5°
Gismeteo
Подробный прогноз
Сегодня
12:00 | Ежедневная синхронизация
Понедельник
15:00 | Статус разработки
Вторник
21:00 | Театр
Показать больше...
Уведомления
Очистить всё
Выход
v.ivanov@omp.ru
Александр Федоров
Новости компании
Языковая школа
Показать больше... +21
Система
Снимок экрана сохранён
Снимок экрана сохранён
AQUARIUS
100%
4G
11:15
Вторник
30 нояб.
6°
AQUARIUS
Телефон
Сообщения
Браузер
Камера
Контакты
Галерея
Настройки
Файлы
Календарь
Помощь
Часы
Почта
Калькулятор
Медиа
Диктофон
Офис
Погода
Заметки
AQUARIUS

Аврора ОС

- Базируется на Linux
- Начало было положено в 2016 году на базе Sailfish OS
- С 2018 года является полностью независимым от Sailfish OS проектом,
  - есть все права и лицензии
  - команда, кодовая база полностью в РФ
  - всё собирается и развивается здесь

Дорожная карта

1. Аврора ОС
2. Инструменты разработчика
3. CLEAN архитектура и Coordinator
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. DI
8. Итоги

Aurora IDE

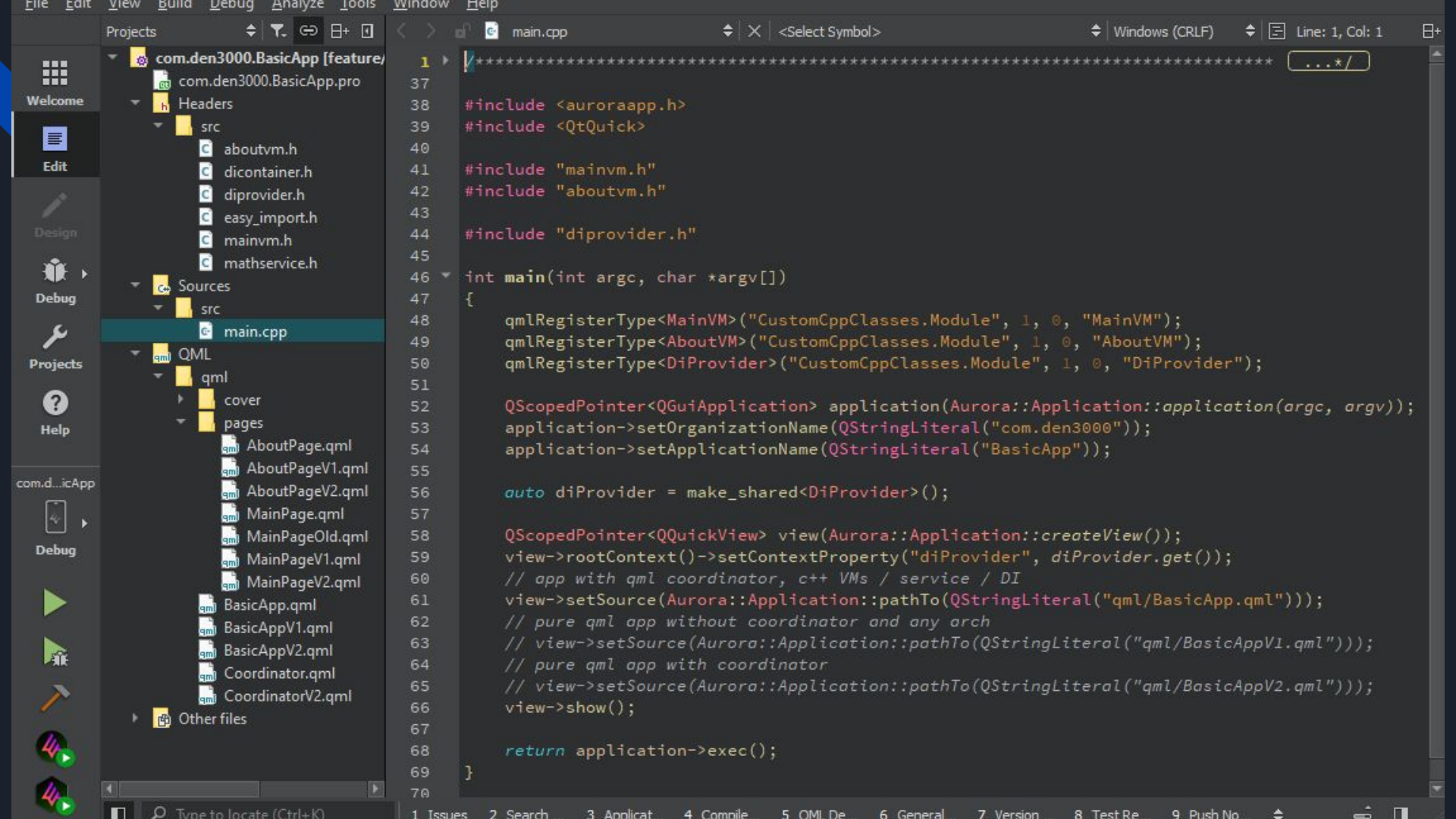
- Qt - Кроссплатформенный фреймворк
- На Аврора ОС доступна версия Qt 5.6
- C++ / QML
- Позволяет получить доступ ко всем системным API
- Набор UI компонентов Silica
- Signals and Slots
- Полуавтоматическое управления памятью

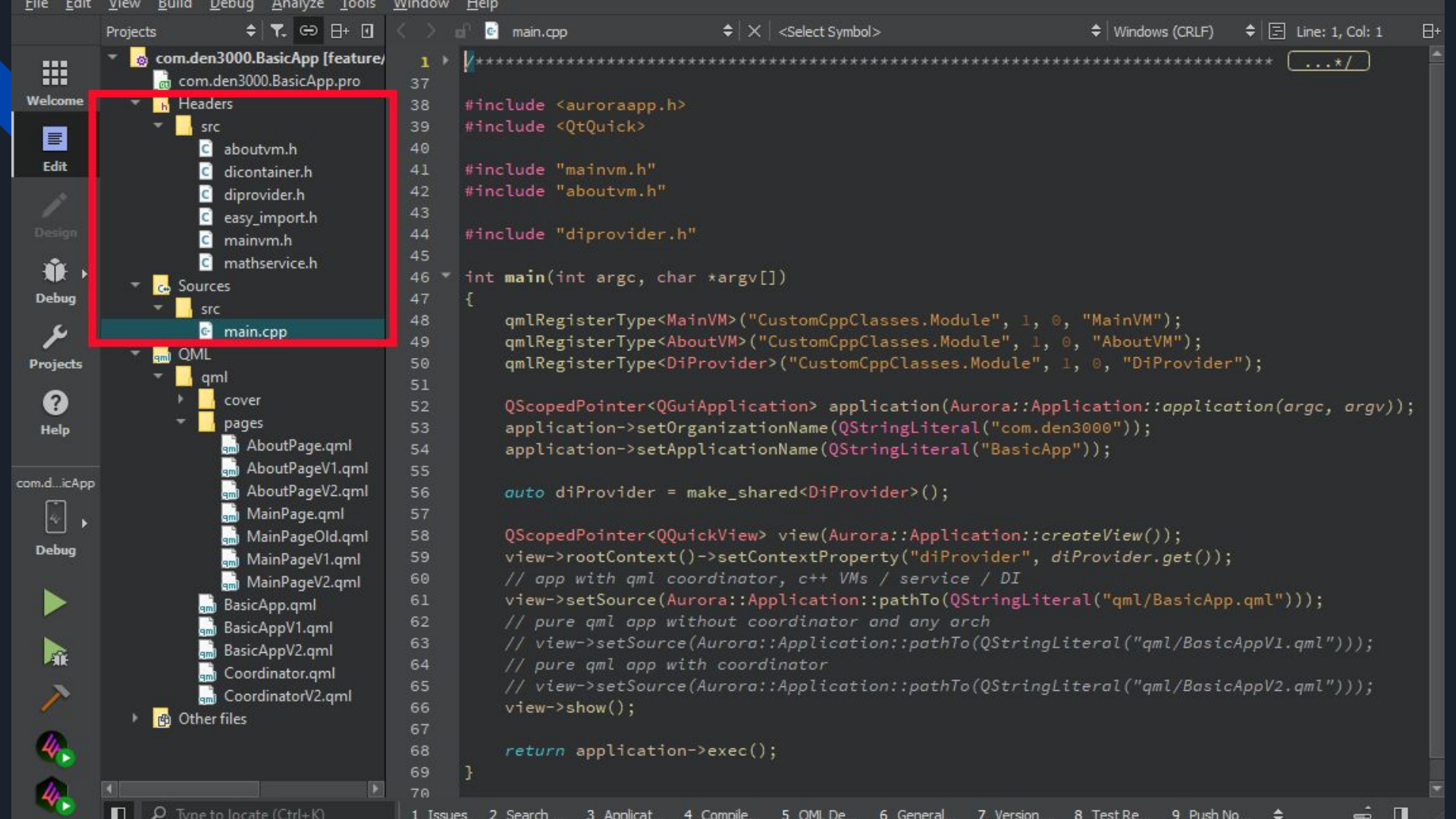
C++

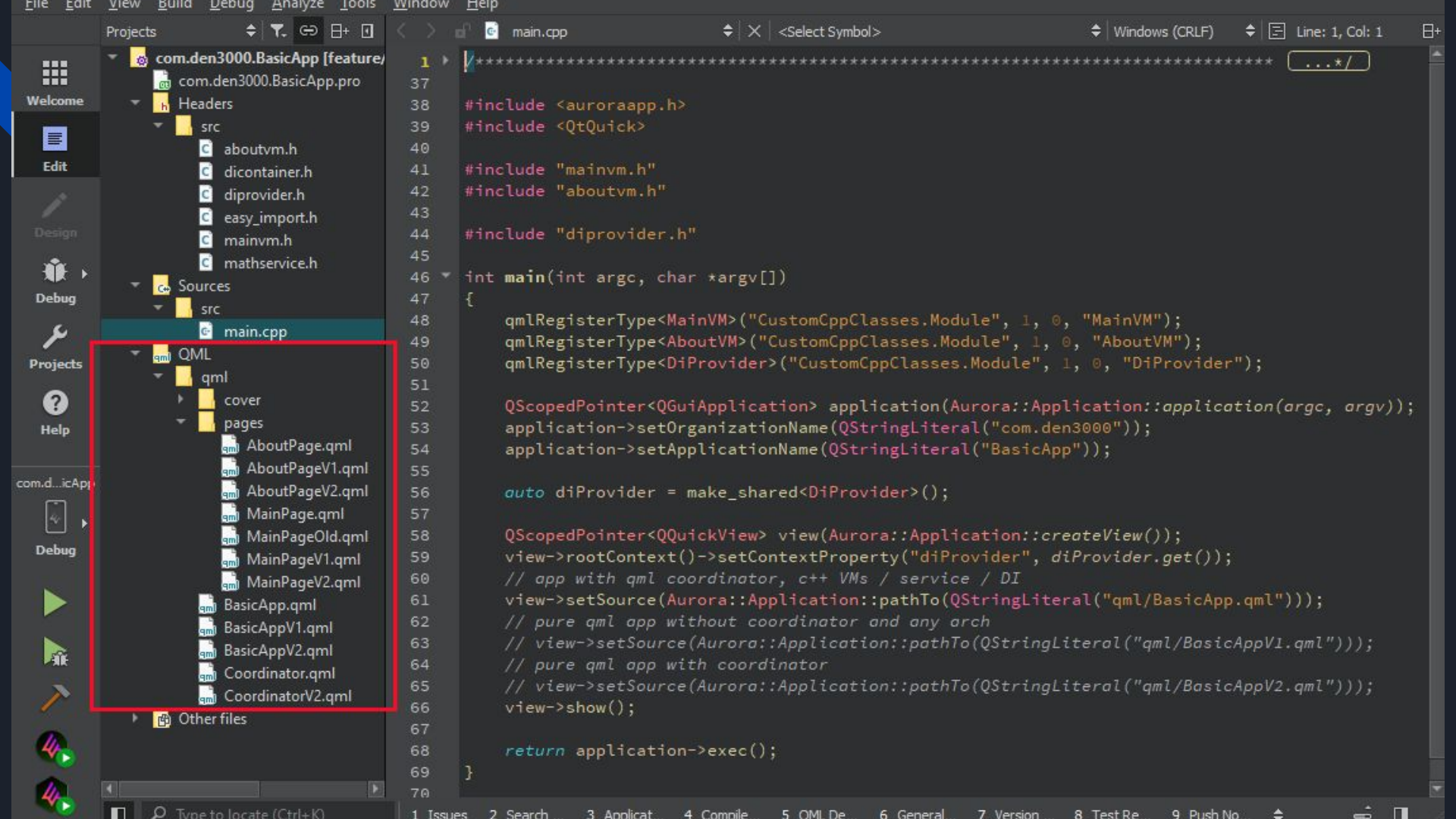
- По умолчанию C++ 14
- Есть возможность использовать C++ 17
- Умные указатели
- Лямбды
- Оказалось не так больно, как изначально ожидалось

QML

- Язык разметки, средой выполнения JavaScript которого является пользовательский движок V4
- Декларативный UI
- Meta Object Compiler
- Полуавтоматическое управления памятью


File Edit View Build Debug Analyze Tools Window Help
Projects
com.den3000.BasicApp [feature/
com.den3000.BasicApp.pro
Headers
src
aboutvm.h
dicontainer.h
diprovider.h
easy_import.h
mainvm.h
mathservice.h
Sources
src
main.cpp
QML
qml
cover
pages
AboutPage.qml
AboutPageV1.qml
AboutPageV2.qml
MainPage.qml
MainPageOld.qml
MainPageV1.qml
MainPageV2.qml
BasicApp.qml
BasicAppV1.qml
BasicAppV2.qml
Coordinator.qml
CoordinatorV2.qml
Other files
Welcome
Edit
Design
Debug
Projects
Help
com.d...icApp
Debug
main.cpp
<Select Symbol>
Windows (CRLF)
Line: 1, Col: 1
1 /*** .../
37
38 #include <auroraapp.h>
39 #include <QtQuick>
40
41 #include "mainvm.h"
42 #include "aboutvm.h"
43
44 #include "diprovider.h"
45
46 int main(int argc, char *argv[])
47 {
48 qmlRegisterType<MainVM>("CustomCppClasses.Module", 1, 0, "MainVM");
49 qmlRegisterType<AboutVM>("CustomCppClasses.Module", 1, 0, "AboutVM");
50 qmlRegisterType<DiProvider>("CustomCppClasses.Module", 1, 0, "DiProvider");
51
52 QScopedPointer<QGuiApplication> application(Aurora::Application::application(argc, argv));
53 application->setOrganizationName(QStringLiteral("com.den3000"));
54 application->setApplicationName(QStringLiteral("BasicApp"));
55
56 auto diProvider = make_shared<DiProvider>();
57
58 QScopedPointer<QQuickView> view(Aurora::Application::createView());
59 view->rootContext()->setContextProperty("diProvider", diProvider.get());
60 // app with qml coordinator, c++ VMs / service / DI
61 view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicApp.qml")));
62 // pure qml app without coordinator and any arch
63 // view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicAppV1.qml")));
64 // pure qml app with coordinator
65 // view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicAppV2.qml")));
66 view->show();
67
68 return application->exec();
69 }
70
Type to locate (Ctrl+K)
1 Issues 2 Search 3 Applicat 4 Compile 5 QML De 6 General 7 Version 8 Test Re 9 Push No

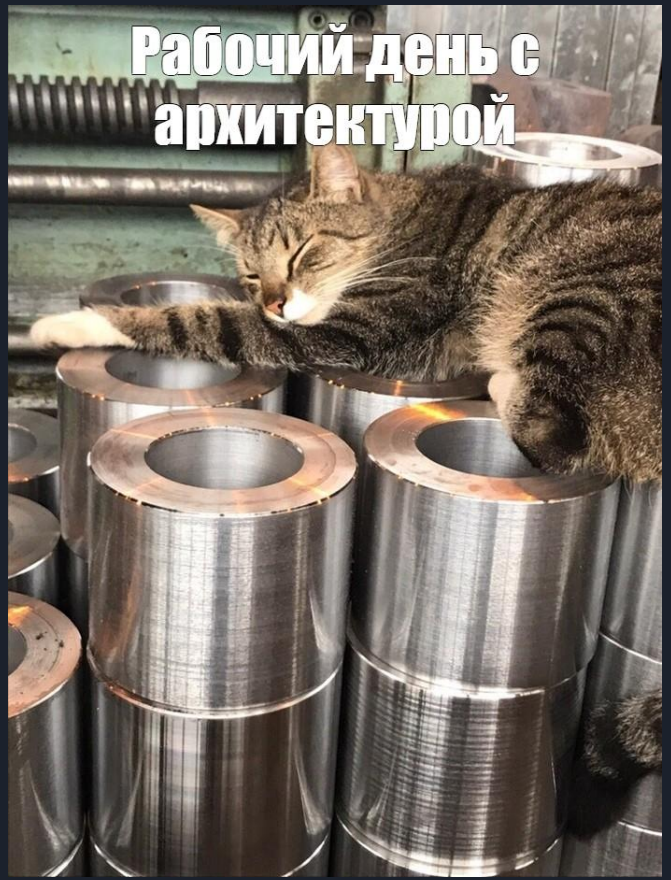

File Edit View Build Debug Analyze Tools Window Help
Projects
com.den3000.BasicApp [feature/
com.den3000.BasicApp.pro
Headers
src
aboutvm.h
dicontainer.h
diprovider.h
easy_import.h
mainvm.h
mathservice.h
Sources
src
main.cpp
QML
qml
cover
pages
AboutPage.qml
AboutPageV1.qml
AboutPageV2.qml
MainPage.qml
MainPageOld.qml
MainPageV1.qml
MainPageV2.qml
BasicApp.qml
BasicAppV1.qml
BasicAppV2.qml
Coordinator.qml
CoordinatorV2.qml
Other files
Welcome
Edit
Design
Debug
Projects
Help
com.d...icApp
Debug
main.cpp
<Select Symbol>
Windows (CRLF)
Line: 1, Col: 1
/*****...*/
#include <auroraapp.h>
#include <QtQuick>
#include "mainvm.h"
#include "aboutvm.h"
#include "diprovider.h"
int main(int argc, char *argv[])
{
qmlRegisterType<MainVM>("CustomCppClasses.Module", 1, 0, "MainVM");
qmlRegisterType<AboutVM>("CustomCppClasses.Module", 1, 0, "AboutVM");
qmlRegisterType<DiProvider>("CustomCppClasses.Module", 1, 0, "DiProvider");
QScopedPointer<QGuiApplication> application(Aurora::Application::application(argc, argv));
application->setOrganizationName(QStringLiteral("com.den3000"));
application->setApplicationName(QStringLiteral("BasicApp"));
auto diProvider = make_shared<DiProvider>();
QScopedPointer<QQuickView> view(Aurora::Application::createView());
view->rootContext()->setContextProperty("diProvider", diProvider.get());
// app with qml coordinator, c++ VMs / service / DI
view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicApp.qml")));
// pure qml app without coordinator and any arch
// view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicAppV1.qml")));
// pure qml app with coordinator
// view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicAppV2.qml")));
view->show();
return application->exec();
}
Type to locate (Ctrl+K)
1 Issues 2 Search 3 Applicat 4 Compile 5 QML De 6 General 7 Version 8 Test Re 9 Push No


File Edit View Build Debug Analyze Tools Window Help
Projects
com.den3000.BasicApp [feature/
com.den3000.BasicApp.pro
Headers
src
aboutvm.h
dicontainer.h
diprovider.h
easy_import.h
mainvm.h
mathservice.h
Sources
src
main.cpp
QML
qml
cover
pages
AboutPage.qml
AboutPageV1.qml
AboutPageV2.qml
MainPage.qml
MainPageOld.qml
MainPageV1.qml
MainPageV2.qml
BasicApp.qml
BasicAppV1.qml
BasicAppV2.qml
Coordinator.qml
CoordinatorV2.qml
Other files
Welcome
Edit
Design
Debug
Projects
Help
Debug
main.cpp
<Select Symbol>
Windows (CRLF)
Line: 1, Col: 1
1 /*****...*/
37
38 #include <auroraapp.h>
39 #include <QtQuick>
40
41 #include "mainvm.h"
42 #include "aboutvm.h"
43
44 #include "diprovider.h"
45
46 int main(int argc, char *argv[])
47 {
48 qmlRegisterType<MainVM>("CustomCppClasses.Module", 1, 0, "MainVM");
49 qmlRegisterType<AboutVM>("CustomCppClasses.Module", 1, 0, "AboutVM");
50 qmlRegisterType<DiProvider>("CustomCppClasses.Module", 1, 0, "DiProvider");
51
52 QScopedPointer<QGuiApplication> application(Aurora::Application::application(argc, argv));
53 application->setOrganizationName(QStringLiteral("com.den3000"));
54 application->setApplicationName(QStringLiteral("BasicApp"));
55
56 auto diProvider = make_shared<DiProvider>();
57
58 QScopedPointer<QQuickView> view(Aurora::Application::createView());
59 view->rootContext()->setContextProperty("diProvider", diProvider.get());
60 // app with qml coordinator, c++ VMs / service / DI
61 view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicApp.qml")));
62 // pure qml app without coordinator and any arch
63 // view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicAppV1.qml")));
64 // pure qml app with coordinator
65 // view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicAppV2.qml")));
66 view->show();
67
68 return application->exec();
69 }
70
Type to locate (Ctrl+K)
1 Issues 2 Search 3 Applicat 4 Compile 5 QML De 6 General 7 Version 8 Test Re 9 Push No

Дорожная карта

1. Аврора ОС
2. Qt/C++/QML
3. CLEAN архитектура и Coordinator
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. DI
8. Итоги


Рабочий день без архитектуры


Рабочий день с архитектурой

CLEAN архитектура и Координатор

- Каждый экран в приложении должен быть независимой сущностью
- В каждом экране должно быть разделение между UI и бизнес-логикой
- Необходим подход, который позволит отделить логику навигации между экранами от их реализации

CLEAN архитектура и Координатор

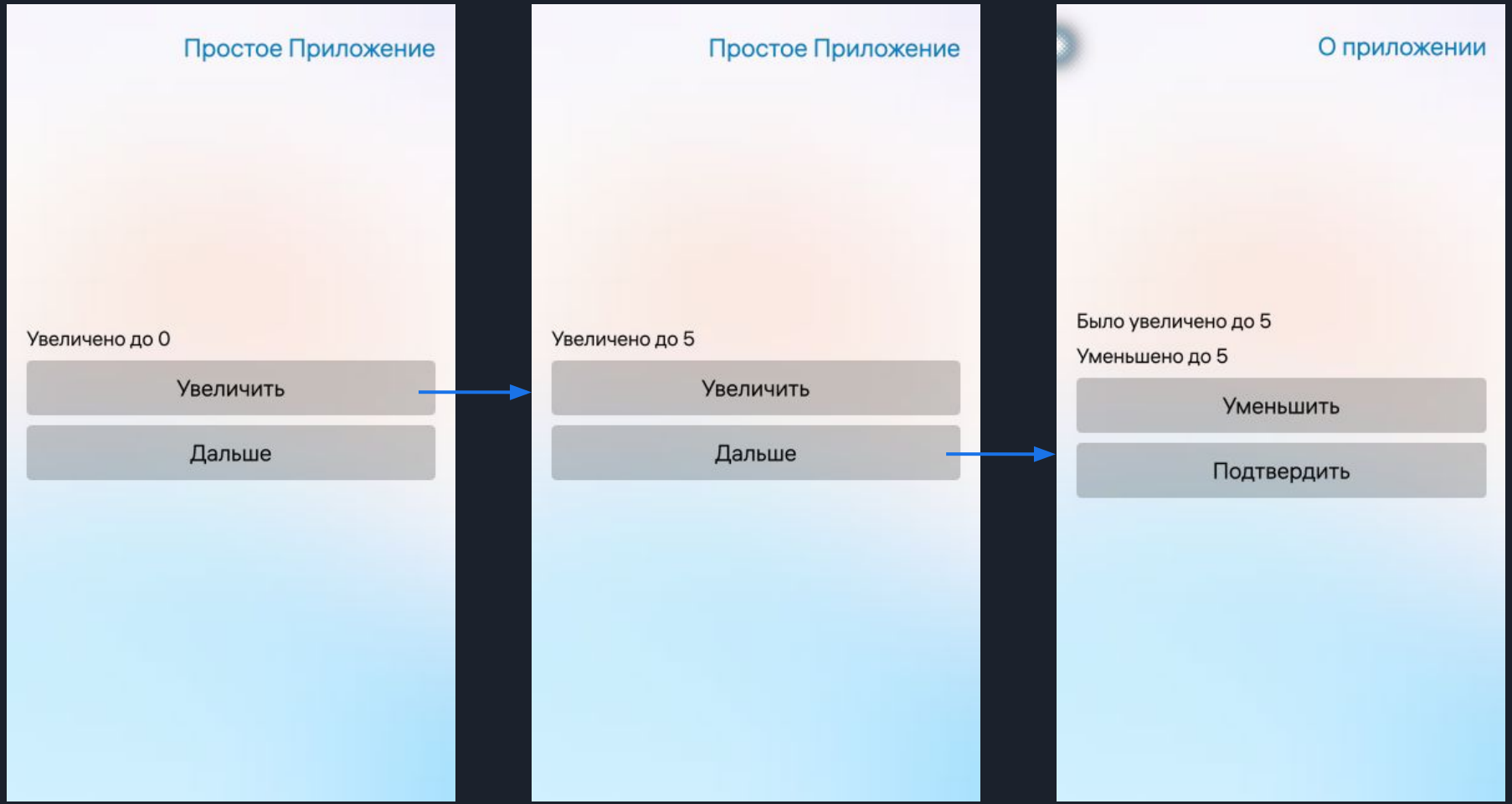
- Бизнес-логика должна быть разделена на слои
  - работа с сетью
  - работа с БД
  - работа с файлами
  - работа с шифрованием (если оно есть)
- преобразование моделей при пересечении границ слоев

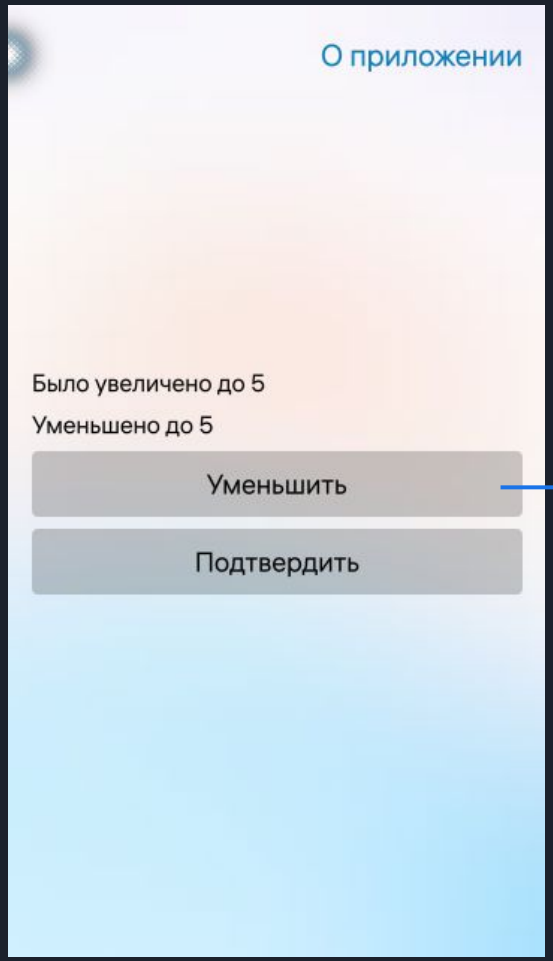
CLEAN архитектура и Координатор

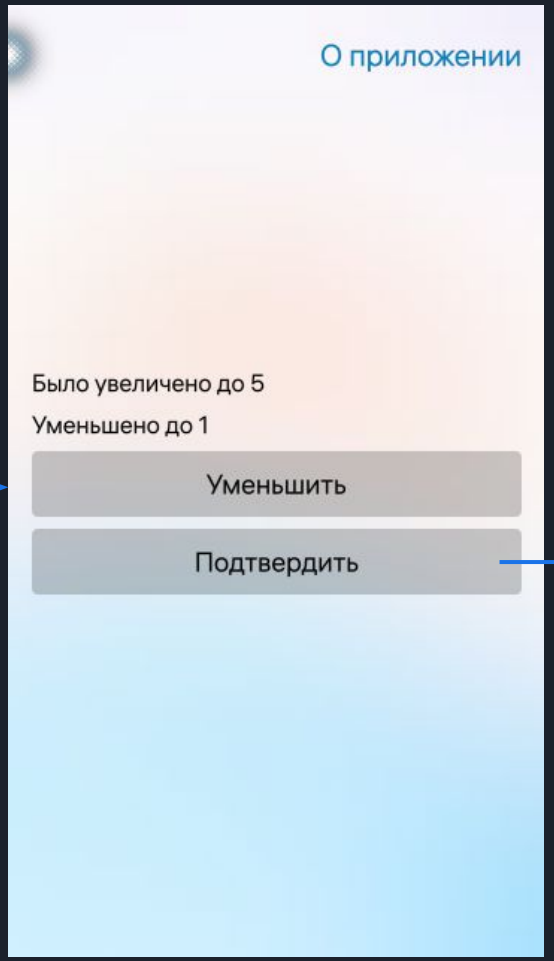
- Чтобы не блокировать UI при выполнении «тяжёлых» операций нужна некоторая “лёгкая” многопоточность
- Наконец нужна система как собирать все выше описанные «кубики» в приложение

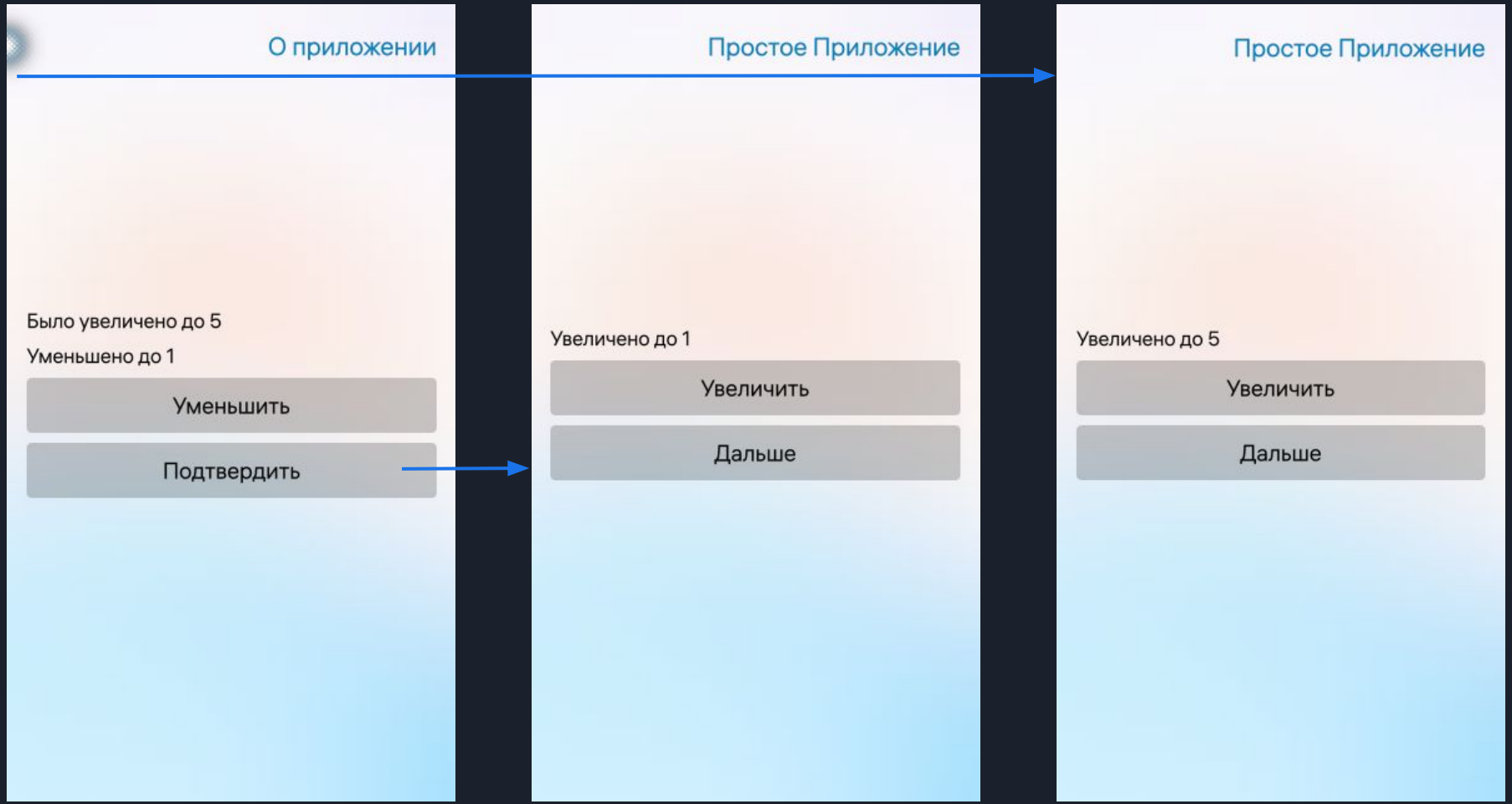
Дорожная карта

1. Аврора ОС
2. Инструменты разработчика
3. CLEAN архитектура и Coordinator
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. DI
8. Итоги


Простое Приложение
Увеличено до 0
Увеличить
Дальше
Простое Приложение
Увеличено до 5
Увеличить
Дальше
О приложении
Было увеличено до 5
Уменьшено до 5
Уменьшить
Подтвердить


О приложении
Было увеличено до 5
Уменьшено до 5
Уменьшить
Подтвердить


О приложении
Было увеличено до 5
Уменьшено до 1
Уменьшить
Подтвердить


О приложении
Было увеличено до 5
Уменьшено до 1
Уменьшить
Подтвердить
Простое Приложение
Увеличено до 1
Увеличить
Дальше
Простое Приложение
Увеличено до 5
Увеличить
Дальше

Простое приложение

- 2 экрана
- Каждый экран интерактивен
- Есть возможность передать параметр с первого экрана на второй
- Есть возможность передать результат со второго экрана на первый

C++

main.cpp

```cpp
#include <auroraapp.h>
#include <QtQuick>

int main(int argc, char *argv[])
{
    QScopedPointer<QGuiApplication> application(Aurora::Application::application(argc, argv));
    application->setOrganizationName(QStringLiteral("com.den3000"));
    application->setApplicationName(QStringLiteral("BasicApp"));

    QScopedPointer<QQuickView> view(Aurora::Application::createView());
    view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicApp.qml")));
    view->show();

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

C++

main.cpp

```cpp
#include <auroraapp.h>
#include <QtQuick>

int main(int argc, char *argv[])
{
    QScopedPointer<QGuiApplication> application(Aurora::Application::application(argc, argv));
    application->setOrganizationName(QStringLiteral("com.den3000"));
    application->setApplicationName(QStringLiteral("BasicApp"));

    QScopedPointer<QQuickView> view(Aurora::Application::createView());
    view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicApp.qml")));
    view->show();

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

C++

main.cpp

```cpp
#include <auroraapp.h>
#include <QtQuick>

int main(int argc, char *argv[])
{
    QScopedPointer<QGuiApplication> application(Aurora::Application::application(argc, argv));
    application->setOrganizationName(QStringLiteral("com.den3000"));
    application->setApplicationName(QStringLiteral("BasicApp"));

    QScopedPointer<QQuickView> view(Aurora::Application::createView());
    view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicApp.qml")));
    view->show();

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

QML

BasicApp.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

ApplicationWindow {
    objectName: "applicationWindow"
    initialPage: Qt.resolvedUrl("pages/MainPage.qml")
    cover: Qt.resolvedUrl("cover/DefaultCoverPage.qml")
    allowedOrientations: defaultAllowedOrientations
}
```

QML

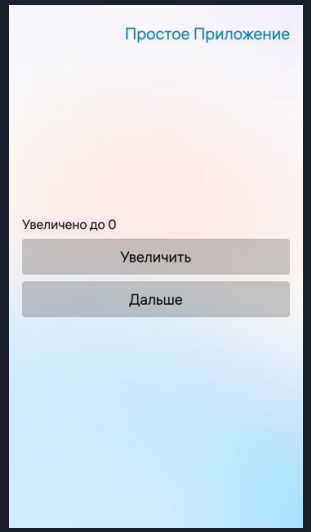
BasicApp.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

ApplicationWindow {
    objectName: "applicationWindow"
    initialPage: Qt.resolvedUrl("pages/MainPage.qml")
    cover: Qt.resolvedUrl("cover/DefaultCoverPage.qml")
    allowedOrientations: defaultAllowedOrientations
}
```

QML

MainPage.qml

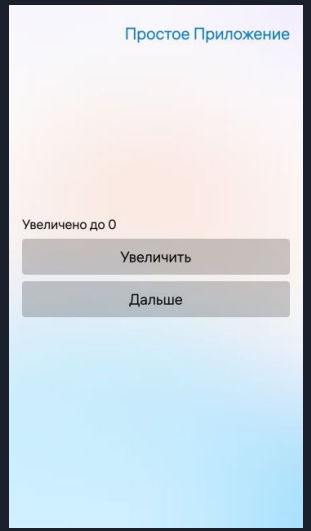
```
Page {
    property int counter: 0

    allowedOrientations: Orientation.All

    PageHeader { title: qsTr("Простое Приложение") }

    Column {
        id: layout
        width: parent.width
        spacing: 16
        anchors.centerIn: parent

        // ...
    }
}
```



MainPage.qml

```
Page {
    property int counter: 0

    allowedOrientations: Orientation.All

    PageHeader { title: qsTr("Простое Приложение") }

    Column {
        id: layout
        width: parent.width
        spacing: 16
        anchors.centerIn: parent

        // ...
    }
}
```

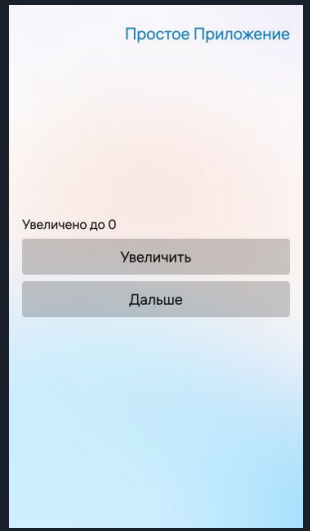
QML



QML

MainPage.qml

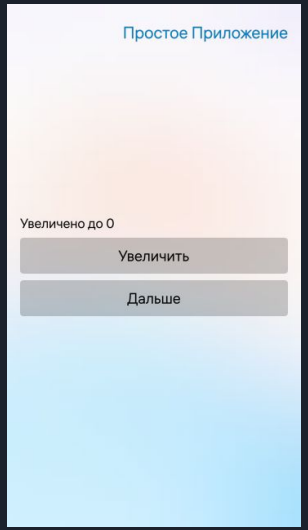
```
Text {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    id: txtCounter
    text: qsTr("Увеличено до %1").arg(counter)
}

Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Увеличить")
    onClicked: counter = counter + 1
}

Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Дальше")
    onClicked: // ...
}
```



QML

MainPage.qml

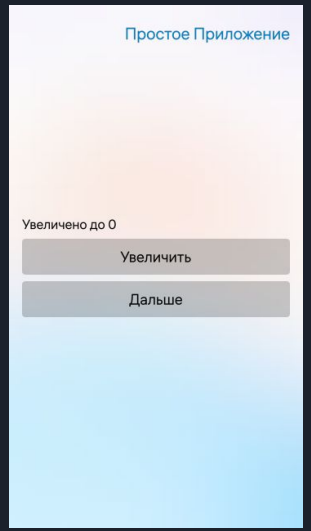
```
Text {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    id: txtCounter
    text: qsTr("Увеличено до %1").arg(counter)
}

Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Увеличить")
    onClicked: counter = counter + 1
}

Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Дальше")
    onClicked: // ...
}
```



QML

MainPage.qml

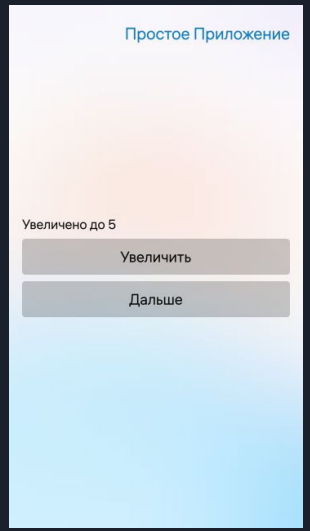
```
Text {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    id: txtCounter
    text: qsTr("Увеличено до %1").arg(counter)
}

Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Увеличить")
    onClicked: counter = counter + 1
}

Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Дальше")
    onClicked: // ...
}
```



QML

MainPage.qml

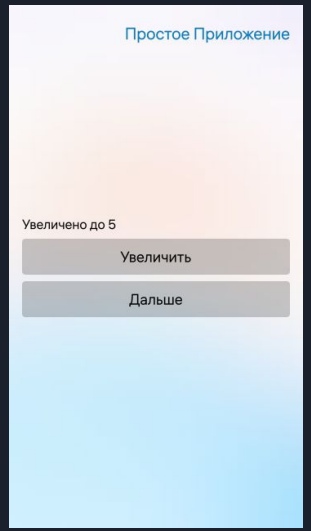
```
Text {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    id: txtCounter
    text: qsTr("Увеличено до %1").arg(counter)
}

Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Увеличить")
    onClicked: counter = counter + 1
}

Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Дальше")
    onClicked: // ...
}
```



QML

MainPage.qml

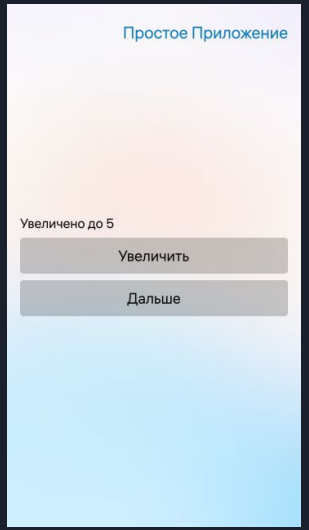
```
Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Дальше")
    onClicked: {
        var page = pageStack.push(Qt.resolvedUrl("AboutPage.qml"), { "inputCounter": counter } )

        var page = pageStack.push(Qt.resolvedUrl("AboutPage.qml"))
        page.inputCounter = counter

        page.onConfirmed.connect(function() {
            counter = page.counter
        })
    }
}
```



MainPage.qml

```
Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Дальше")
    onClicked: {
        var page = pageStack.push(Qt.resolvedUrl("AboutPage.qml"), { "inputCounter": counter } )

        var page = pageStack.push(Qt.resolvedUrl("AboutPage.qml"))
        page.inputCounter = counter

        page.onConfirmed.connect(function() {
            counter = page.counter
        })
    }
}
```

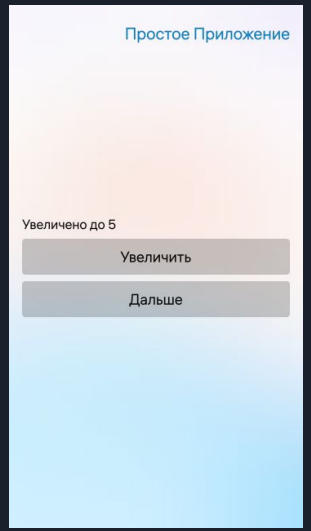
QML



QML

MainPage.qml

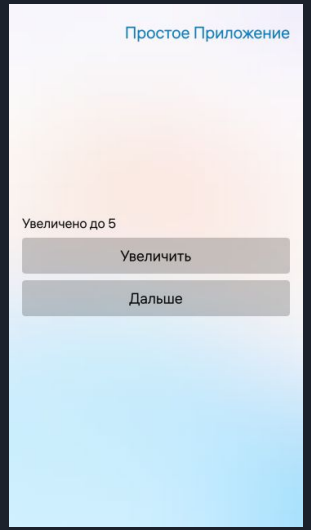
```
Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Дальше")
    onClicked: {
        var page = pageStack.push(Qt.resolvedUrl("AboutPage.qml"), { "inputCounter": counter } )

        var page = pageStack.push(Qt.resolvedUrl("AboutPage.qml"))
        page.inputCounter = counter

        page.onConfirmed.connect(function() {
            counter = page.counter
        })
    }
}
```



MainPage.qml

```
Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Дальше")
    onClicked: {
        var page = pageStack.push(Qt.resolvedUrl("AboutPage.qml"), { "inputCounter": counter } )

        var page = pageStack.push(Qt.resolvedUrl("AboutPage.qml"))
        page.inputCounter = counter

        page.onConfirmed.connect(function() {
            counter = page.counter
        })
    }
}
```

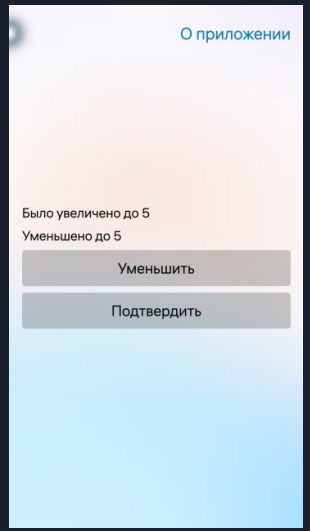
QML



QML

AboutPage.qml

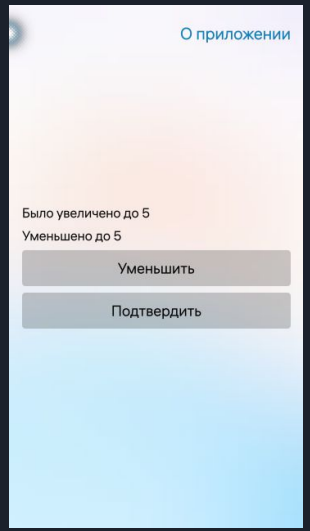
```
Page {
    property int inputCounter
    property int counter: inputCounter

    signal onConfirmed

    allowedOrientations: Orientation.All

    PageHeader { title: qsTr("О приложении") }

    Column {
        id: layout
        width: parent.width
        spacing: 16
        anchors.centerIn: parent
        // ...
    }
}
```



QML

AboutPage.qml

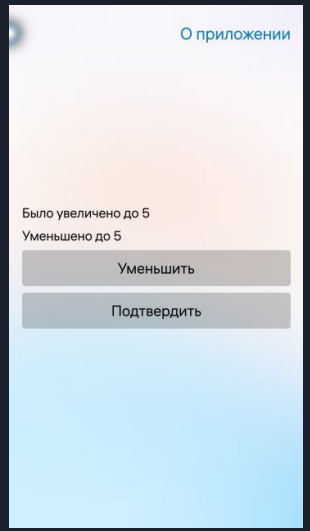
```
Page {
    property int inputCounter
    property int counter: inputCounter

    signal onConfirmed

    allowedOrientations: Orientation.All

    PageHeader { title: qsTr("О приложении") }

    Column {
        id: layout
        width: parent.width
        spacing: 16
        anchors.centerIn: parent
        // ...
    }
}
```



QML

AboutPage.qml

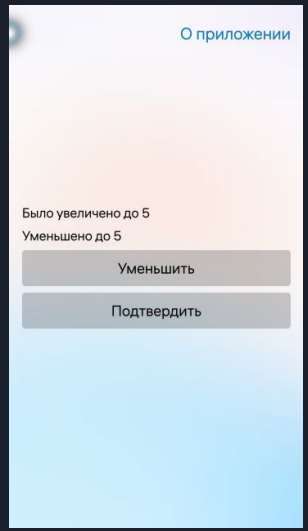
```
Page {
    property int inputCounter
    property int counter: inputCounter

    signal onConfirmed

    allowedOrientations: Orientation.All

    PageHeader { title: qsTr("О приложении") }

    Column {
        id: layout
        width: parent.width
        spacing: 16
        anchors.centerIn: parent
        // ...
    }
}
```



QML

AboutPage.qml

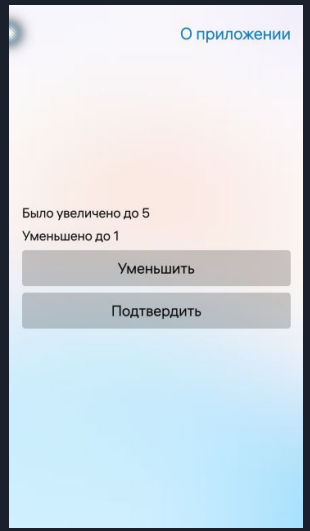
```
Text {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    id: txtOld
    text: qsTr("Было увеличено до %1").arg(inputCounter)
}

Text {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    id: txtCounter
    text: qsTr("Уменьшено до %1").arg(counter)
}

Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Уменьшить")
    onClicked: counter = counter - 1
}
```



QML

AboutPage.qml

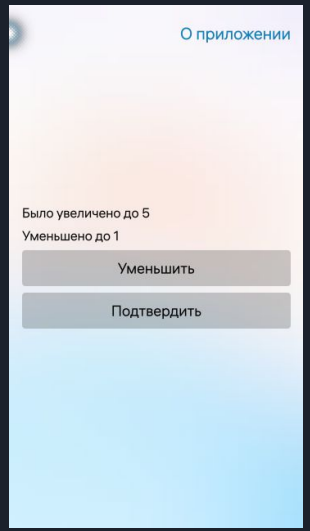
```
Text {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    id: txtOld
    text: qsTr("Было увеличено до %1").arg(inputCounter)
}

Text {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    id: txtCounter
    text: qsTr("Уменьшено до %1").arg(counter)
}

Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Уменьшить")
    onClicked: counter = counter - 1
}
```



QML

AboutPage.qml

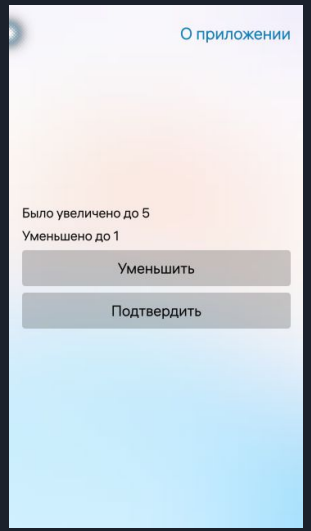
```
Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Подтвердить")
    onClicked: {
        onConfirmed(counter)
        pageStack.pop()
    }
}
```



QML

AboutPage.qml

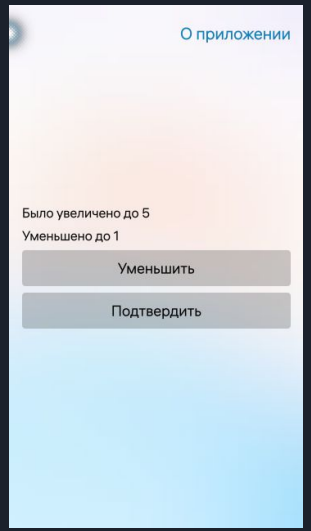
```
Button {
    anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
    text: qsTr("Подтвердить")
    onClicked: {
        onConfirmed(counter)
        pageStack.pop()
    }
}
```



QML

AboutPage.qml - Signals and Slots

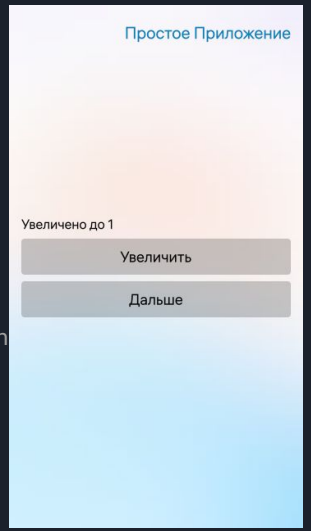
```
Page {
    signal onConfirmed
    Column {
        Button {
            onClicked: {
                onConfirmed(counter)
                pageStack.pop()
            }
        }
    }
}
```



QML

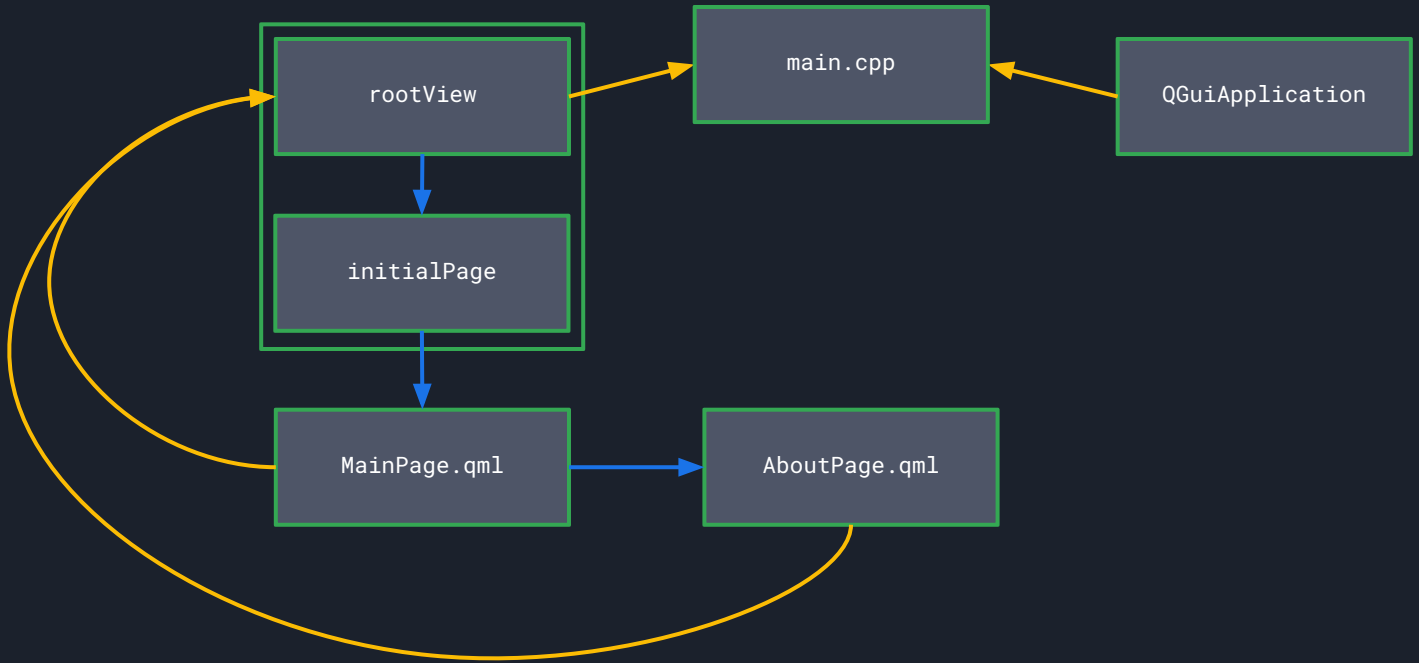
MainPage.qml - Signals and Slots

```
Page {
    Column {
        Button {
            anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin
            text: qsTr("Дальше")
            onClicked: {
                var page = pageStack.push(Qt.resolvedUrl("AboutPage.qml"))
                page.inputCounter = counter

                page.onConfirmed.connect(function() {
                   counter = page.counter
                })
            }
        }
    }
}
```



Coordinator.qml - Signals and Slots

App Flow


rootView
main.cpp
QGuiApplication
initialPage
MainPage.qml
AboutPage.qml

Проблемы простого приложения

- Навигация “вшита” в экраны
- MainPage “знает” слишком много об AboutPage
- Нет возможности динамически выбирать стартовую страницу
- Нет разделения UI и логики

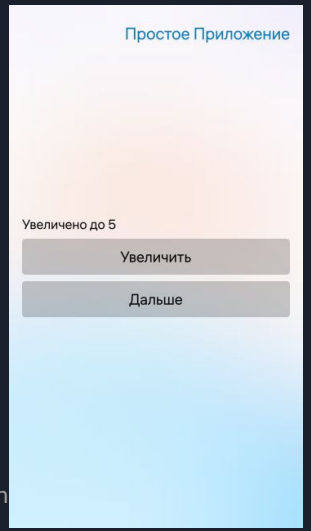
Дорожная карта

1. Аврора ОС
2. Инструменты разработчика
3. CLEAN архитектура и Coordinator
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. DI
8. Итоги

MainPage.qml

```
Page {
    property int counter: 0
    signal onNextPressed(int counter)
    // ...
    Column {
        // ...
        Button {
            anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin
            text: qsTr("Дальше")
            onClicked: onNextPressed(counter)
        }
    }

    function decreased(counterVal) {
       counter = counterVal
    }
}
```

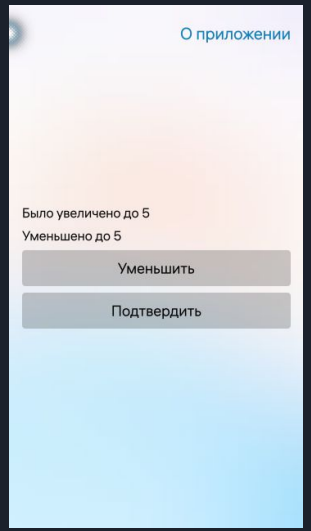
QML



QML

AboutPage.qml

```
Page {
    property int inputCounter
    property int counter: inputCounter
    signal onConfirmed(int counter)
    // ...
    Column {
        // ...

        Button {
            anchors { left: parent.left; right: parent.right; margins: Theme.horizontalPageMargin }
            text: qsTr("Подтвердить")
            onClicked: onConfirmed(counter)
        }
    }
}
```



QML

Coordinator.qml

```
Item {
    property PageStack pageStack

    signal onDecreaseConfirmed(int counter)

    property string mainPage:  "pages/MainPage.qml"
    property string aboutPage: "pages/AboutPage.qml"

    function pushPage(path) { return pageStack.push(Qt.resolvedUrl(path) ) }

    function popPage() { pageStack.pop() }

    function start() { ... }

    function showAbout(counter) { ... }
}
```

QML

Coordinator.qml

```
Item {
    property PageStack pageStack

    signal onDecreaseConfirmed(int counter)

    property string mainPage:  "pages/MainPage.qml"
    property string aboutPage: "pages/AboutPage.qml"

    function pushPage(path) { return pageStack.push(Qt.resolvedUrl(path) ) }

    function popPage() { pageStack.pop() }

    function start() { ... }

    function showAbout(counter) { ... }
}
```

QML

Coordinator.qml

```
Item {
    property PageStack pageStack

    signal onDecreaseConfirmed(int counter)

    property string mainPage:  "pages/MainPage.qml"
    property string aboutPage: "pages/AboutPage.qml"

    function pushPage(path) { return pageStack.push(Qt.resolvedUrl(path) ) }

    function popPage() { pageStack.pop() }

    function start() { ... }

    function showAbout(counter) { ... }
}
```

QML

Coordinator.qml

```
Item {
    property PageStack pageStack

    signal onDecreaseConfirmed(int counter)

    property string mainPage:  "pages/MainPage.qml"
    property string aboutPage: "pages/AboutPage.qml"

    function pushPage(path) { return pageStack.push(Qt.resolvedUrl(path) ) }

    function popPage() { pageStack.pop() }

    function start() { ... }

    function showAbout(counter) { ... }
}
```

QML

Coordinator.qml

```
Item {
    signal onDecreaseConfirmed(int counter)
    // ...
    function start() {
        var page = pushPage(mainPage)
        page.onNextPressed.connect(showAbout)
        onDecreaseConfirmed.connect(page.decreased)
    }

    function showAbout(counter) {
        var page = pushPage(aboutPage)
        page.inputCounter = counter
        page.onConfirmed.connect(function() {
            onDecreaseConfirmed(page.counter)
            popPage()
        })
    }
}
```

QML

Coordinator.qml

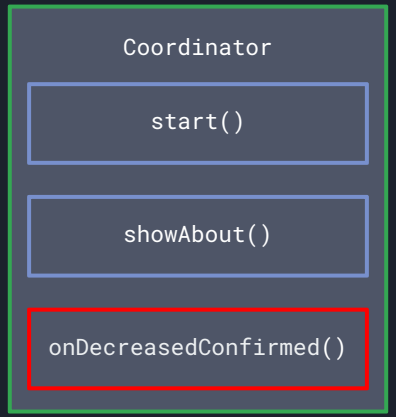
```
Item {
    signal onDecreaseConfirmed(int counter)
    // ...
    function start() {
        var page = pushPage(mainPage)
        page.onNextPressed.connect(showAbout)
        onDecreaseConfirmed.connect(page.decreased)
    }

    function showAbout(counter) {
        var page = pushPage(aboutPage)
        page.inputCounter = counter
        page.onConfirmed.connect(function() {
            onDecreaseConfirmed(page.counter)
            popPage()
        })
    }
}
```

QML

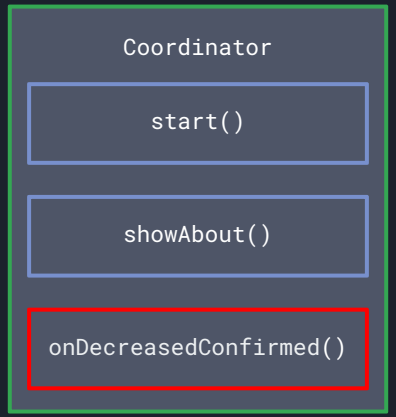
Coordinator.qml

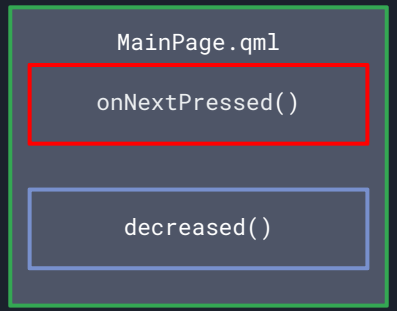
```
Item {
    signal onDecreaseConfirmed(int counter)
    // ...
    function start() {
        var page = pushPage(mainPage)
        page.onNextPressed.connect(showAbout)
        onDecreaseConfirmed.connect(page.decreased)
    }

    function showAbout(counter) {
        var page = pushPage(aboutPage)
        page.inputCounter = counter
        page.onConfirmed.connect(function() {
           onDecreaseConfirmed(page.counter)
           popPage()
        })
    }
}
```

Coordinator.qml - Signals and Slots

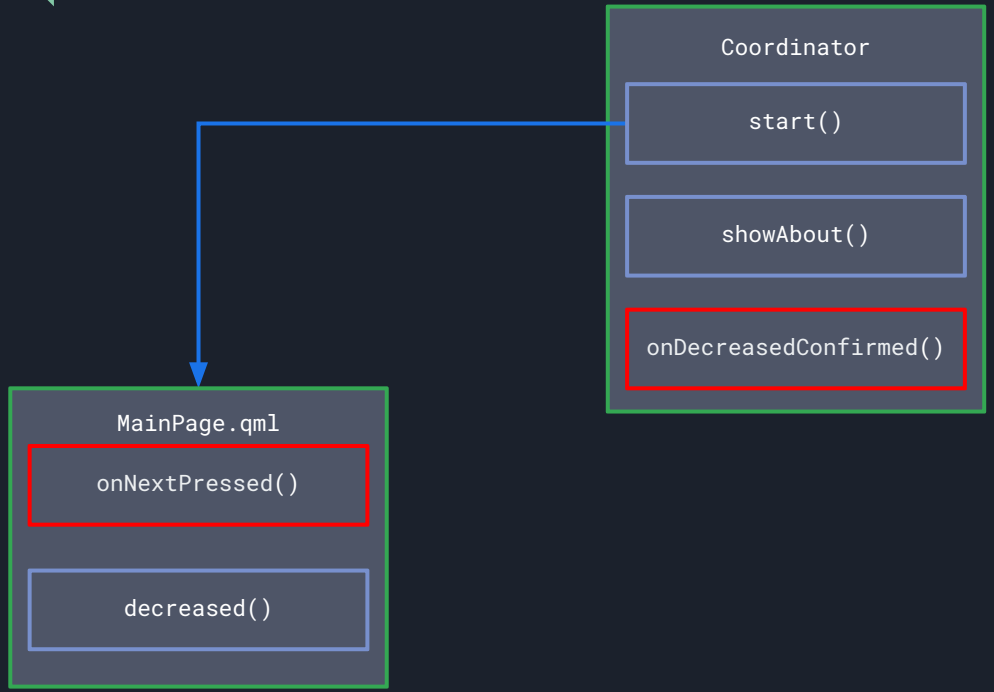


Coordinator.qml - Signals and Slots

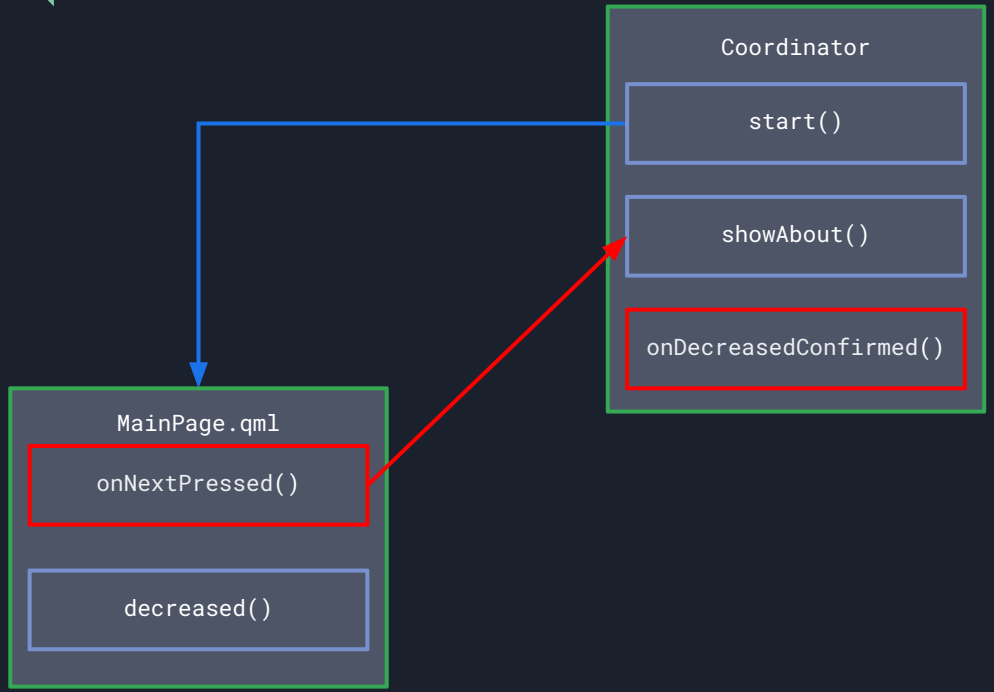




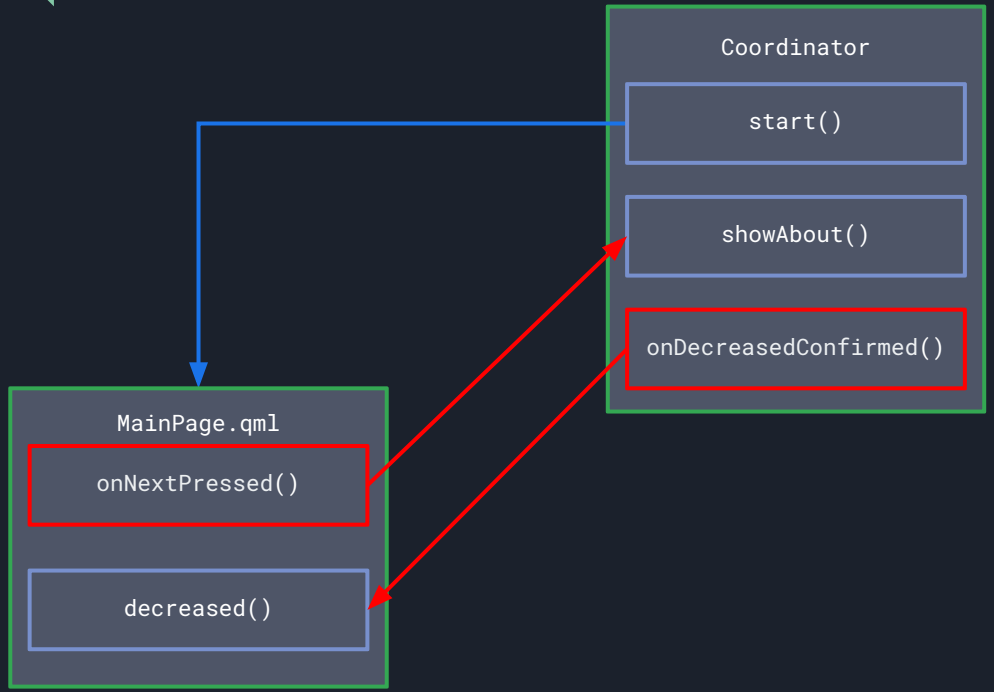
Coordinator.qml - Signals and Slots



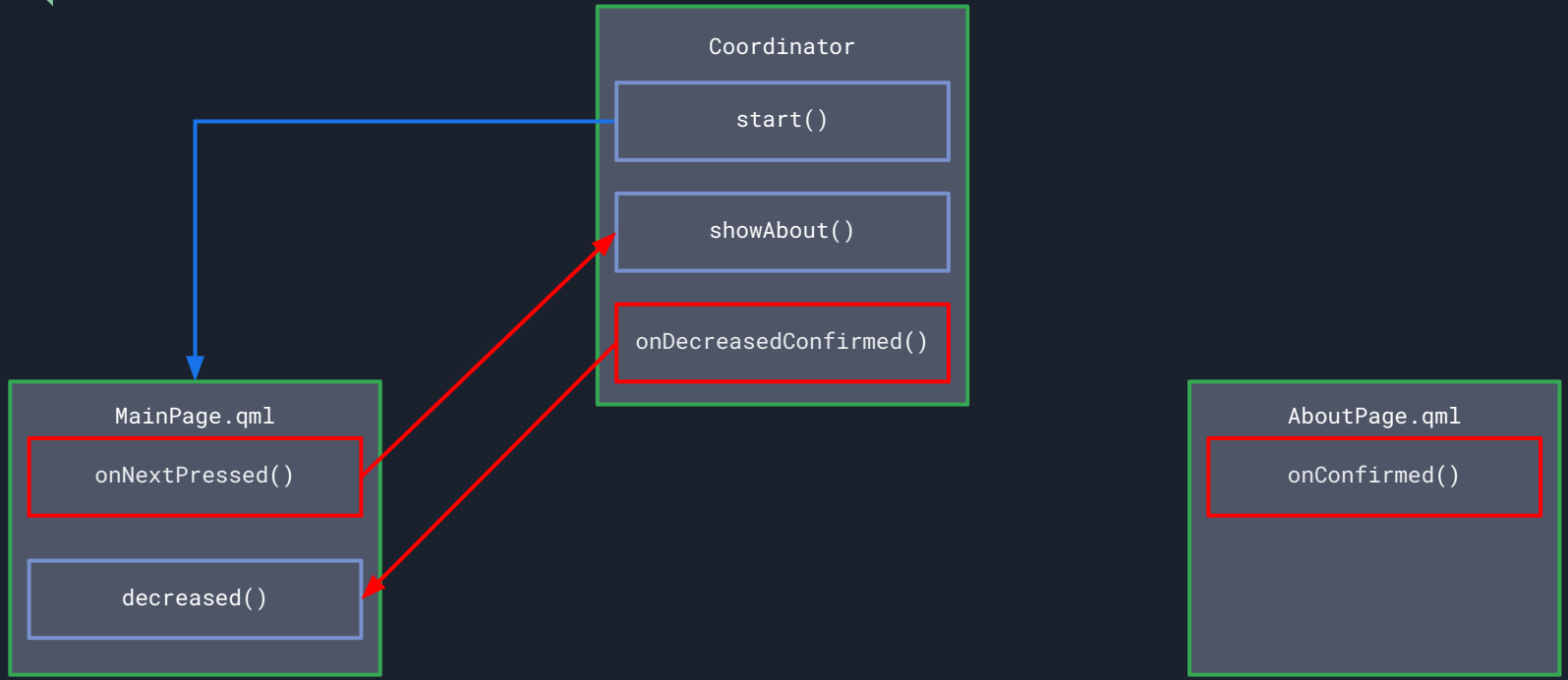
Coordinator.qml - Signals and Slots



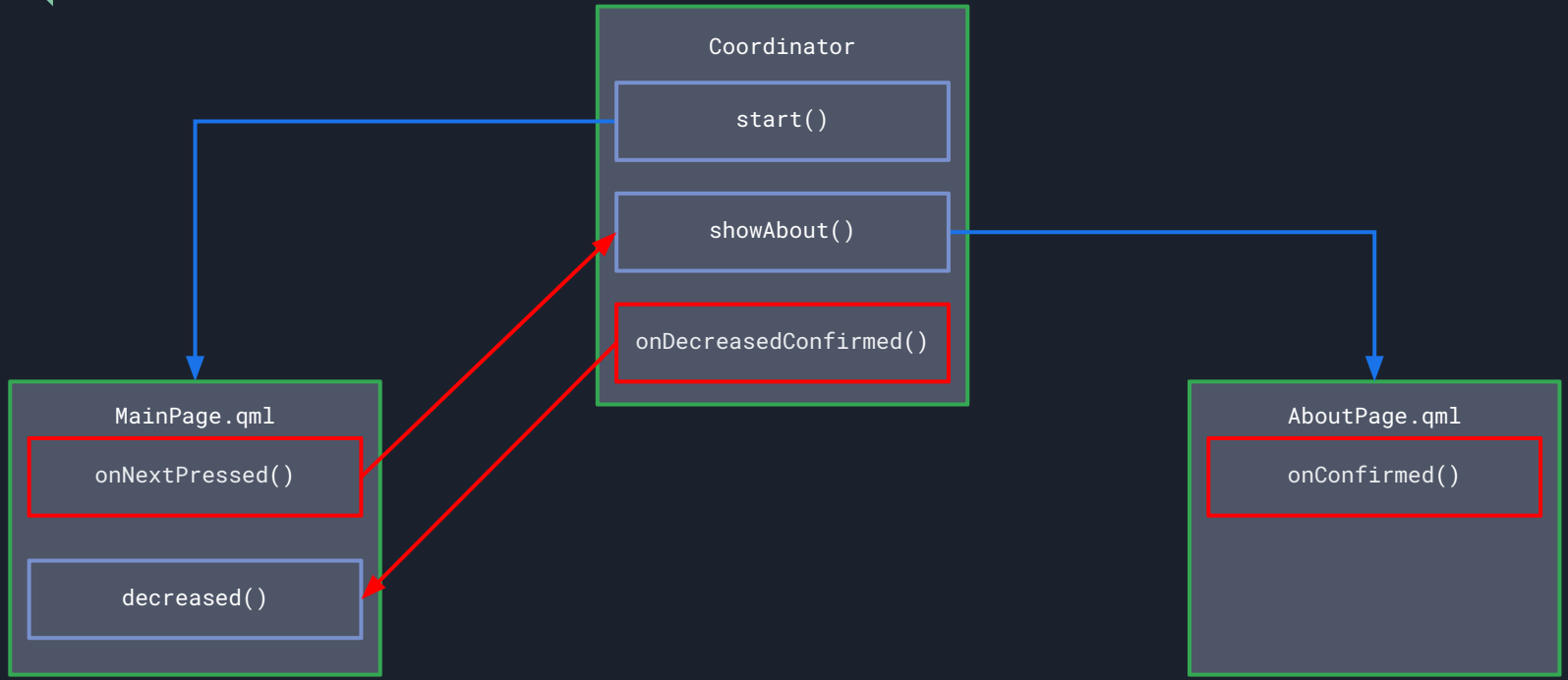
Coordinator.qml - Signals and Slots



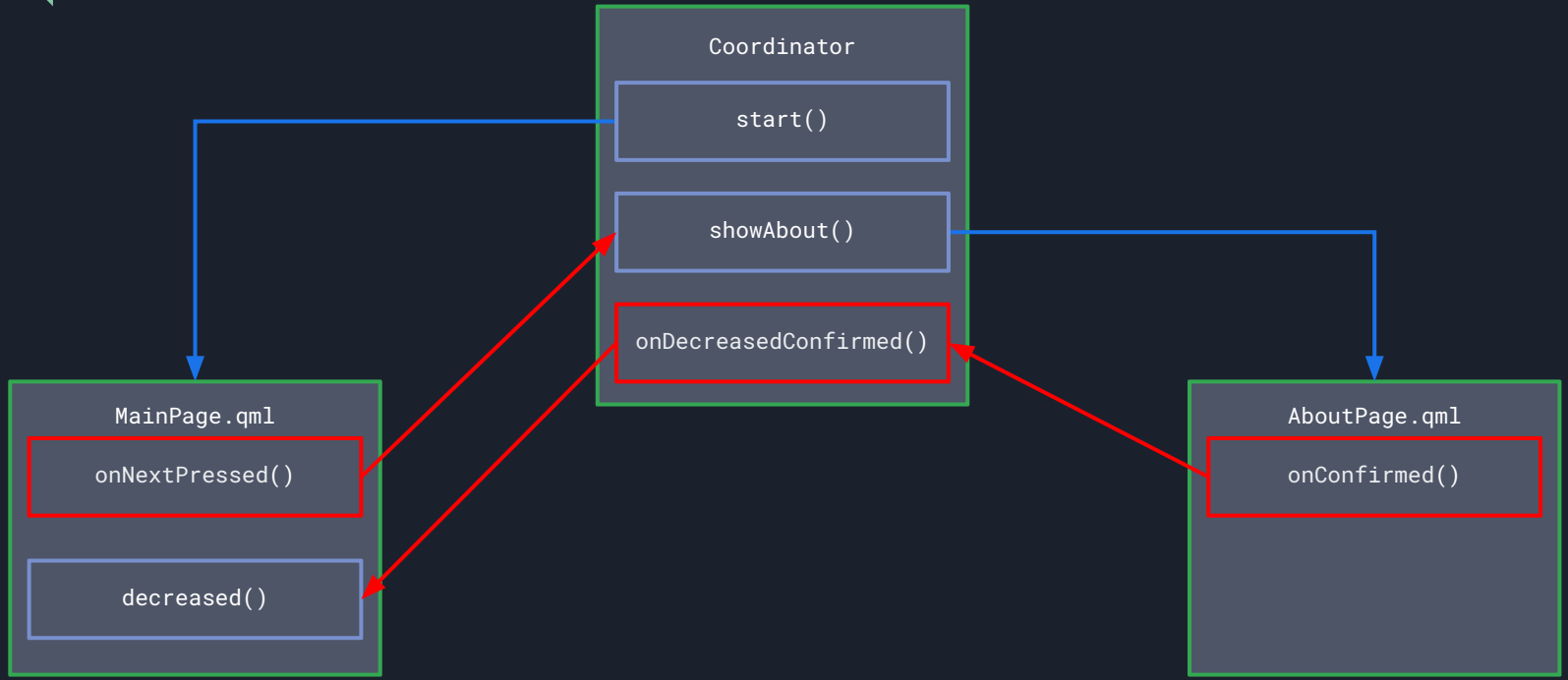
Coordinator.qml - Signals and Slots



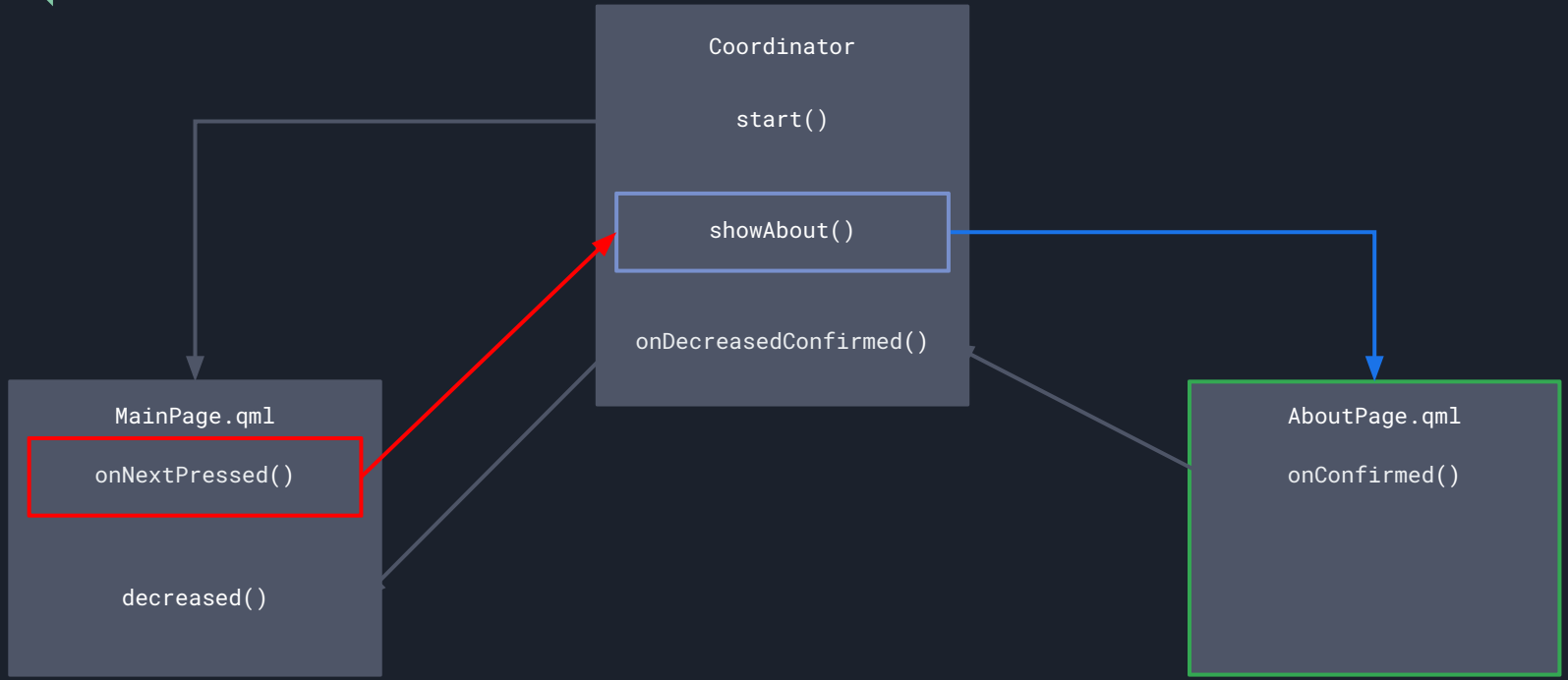
Coordinator.qml - Signals and Slots



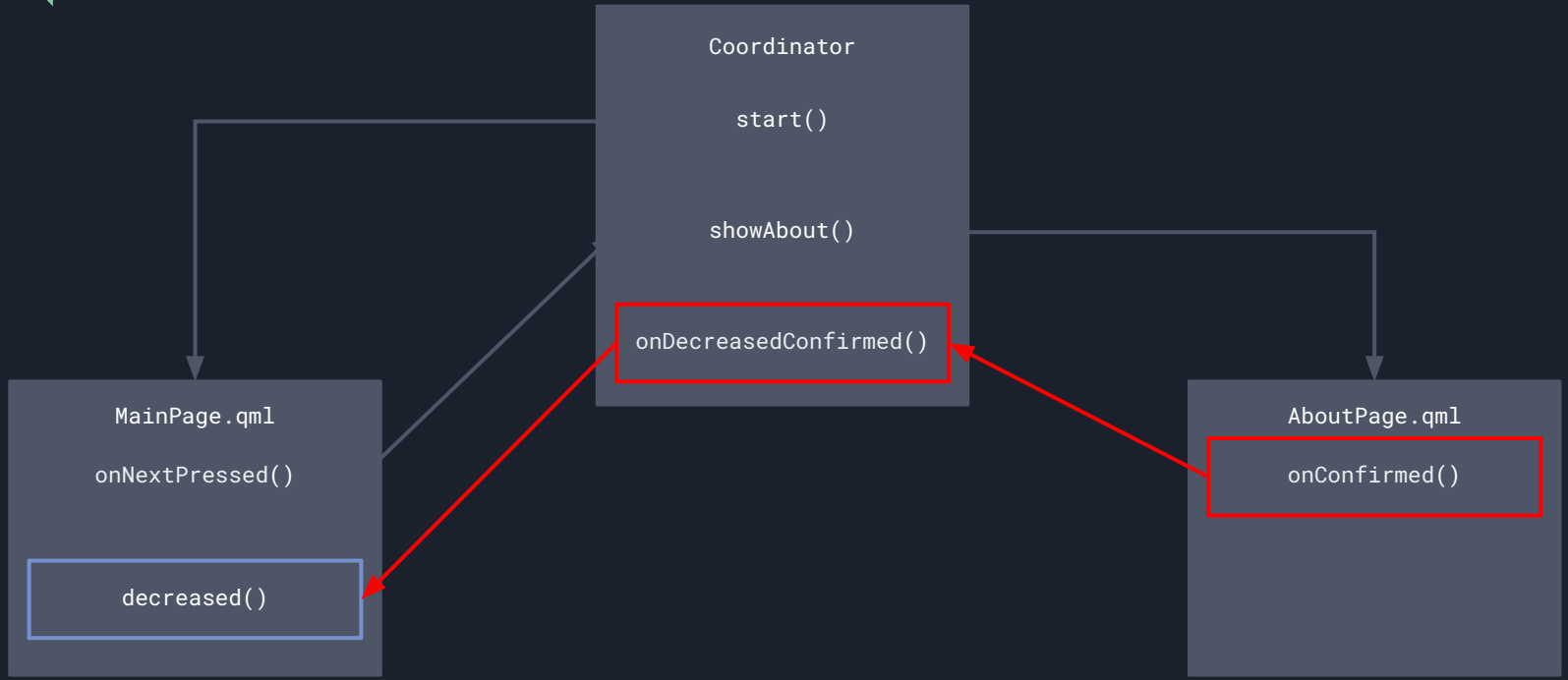
Coordinator.qml - Signals and Slots



Coordinator.qml - Signals and Slots



Coordinator.qml - Signals and Slots



QML

BasicApp.qml

```
ApplicationWindow {
    id: appWindow
    objectName: "applicationWindow"
    cover: Qt.resolvedUrl("cover/DefaultCoverPage.qml")
    allowedOrientations: defaultAllowedOrientations

    Coordinator {
        id: coordinator
        pageStack: appWindow.pageStack
    }

    Component.onCompleted: {
        coordinator.start()
    }
}
```

QML

BasicApp.qml

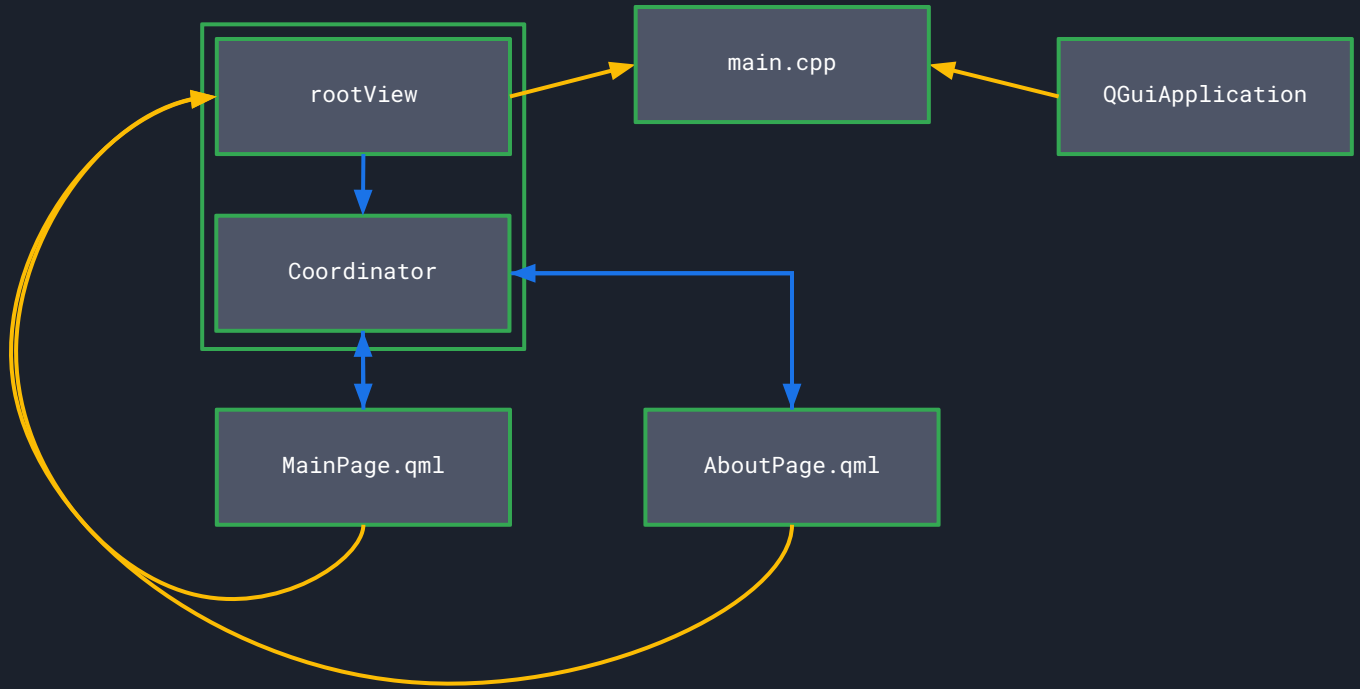
```
ApplicationWindow {
    id: appWindow
    objectName: "applicationWindow"
    cover: Qt.resolvedUrl("cover/DefaultCoverPage.qml")
    allowedOrientations: defaultAllowedOrientations

    Coordinator {
       id: coordinator
       pageStack: appWindow.pageStack
    }

    Component.onCompleted: {
        coordinator.start()
    }
}
```

QML

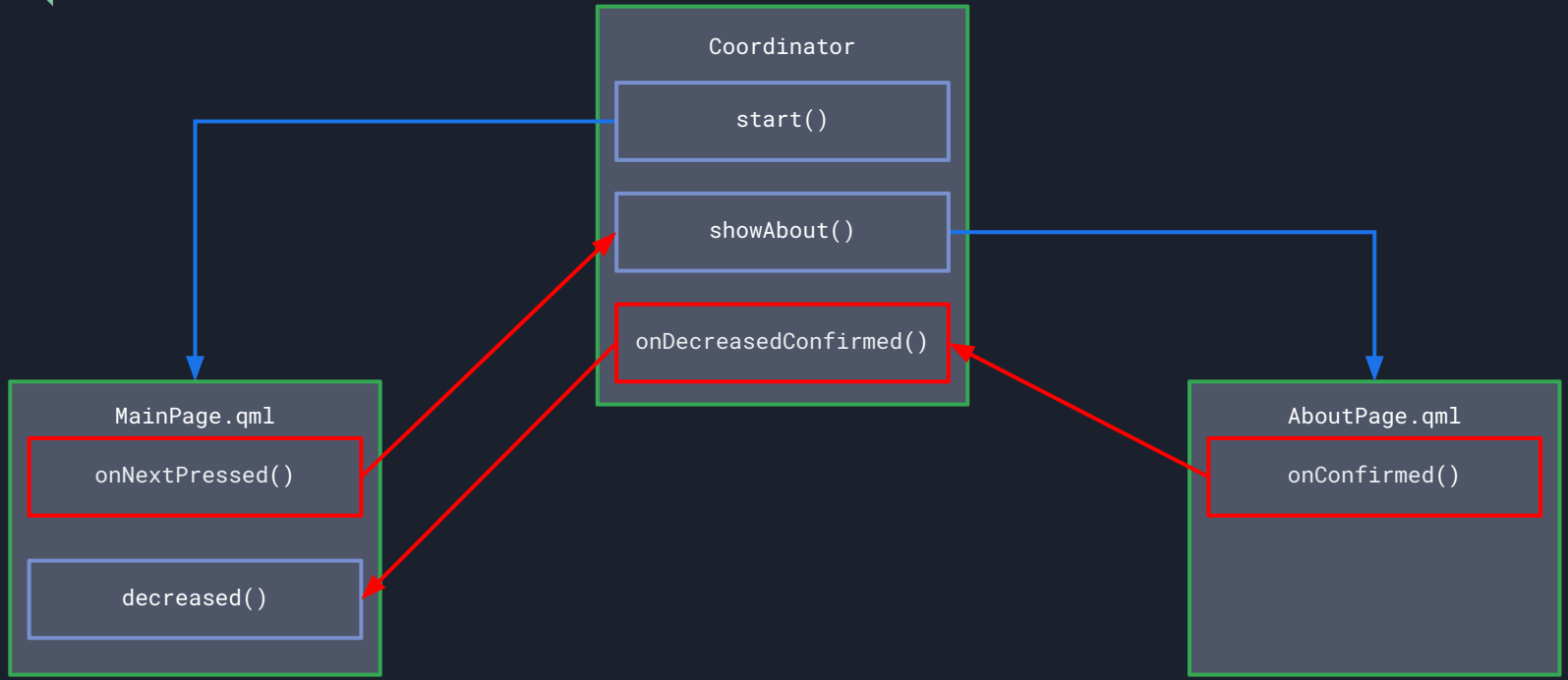
BasicApp.qml

```
ApplicationWindow {
    id: appWindow
    objectName: "applicationWindow"
    cover: Qt.resolvedUrl("cover/DefaultCoverPage.qml")
    allowedOrientations: defaultAllowedOrientations

    Coordinator {
        id: coordinator
        pageStack: appWindow.pageStack
    }

    Component.onCompleted: {
       coordinator.start()
    }
}
```

App Flow


rootView
main.cpp
QGuiApplication
Coordinator
MainPage.qml
AboutPage.qml

Coordinator.qml - Signals and Slots



Проблемы простого приложения

- ~~Навигация “вшита” в экраны~~
- ~~MainPage “знает” слишком много об AboutPage~~
- ~~Нет возможности динамически выбирать стартовую страницу~~
- Нет разделения UI и логики

Дорожная карта

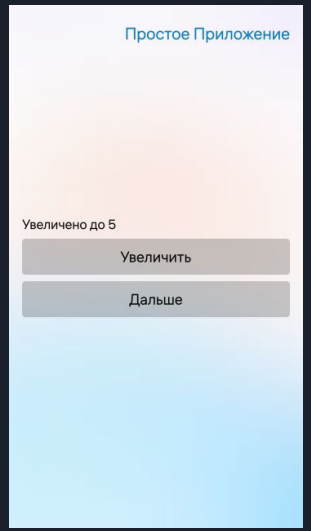
1. Аврора ОС
2. Инструменты разработчика
3. CLEAN архитектура и Coordinator
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. DI
8. Итоги


ViewModel будет не на QML
А на чём?
C++

C++

mainvm.h

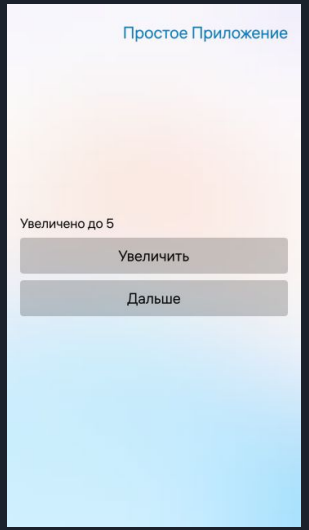
```
class MainVM : public QObject
{
    Q_OBJECT
    // ...

public:
    explicit MainVM(QObject *parent = nullptr): QObject(parent) { qDebug(); };
    ~MainVM() { qDebug(); }

    // ...

signals:
    // ...

public slots:
    // ...
};
```



mainvm.h

```
class MainVM : public QObject
{
    Q_OBJECT
    // ...

public:
    explicit MainVM(QObject *parent = nullptr): QObject(parent) { qDebug(); };
    ~MainVM() { qDebug(); }

    // ...

signals:
    // ...

public slots:
    // ...
};
```

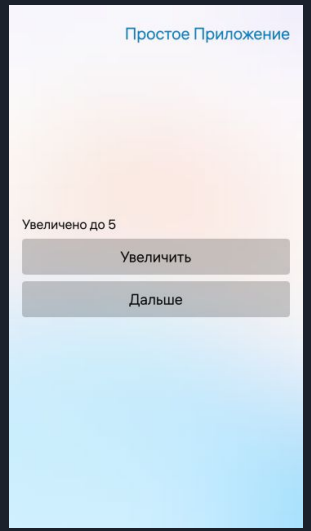
C++



C++

mainvm.h

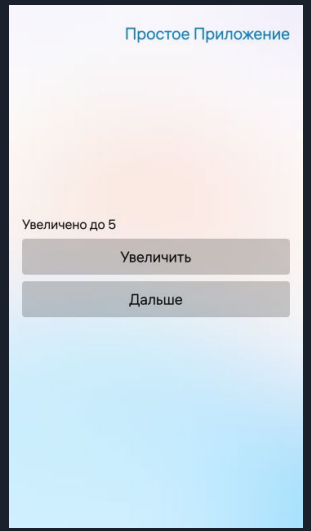
```
class MainVM : public QObject
{
    Q_OBJECT
    // ...

public:
    explicit MainVM(QObject *parent = nullptr): QObject(parent) { qDebug(); };
    ~MainVM() { qDebug(); }

    // ...

signals:
    // ...

public slots:
    // ...
};
```



C++

mainvm.h

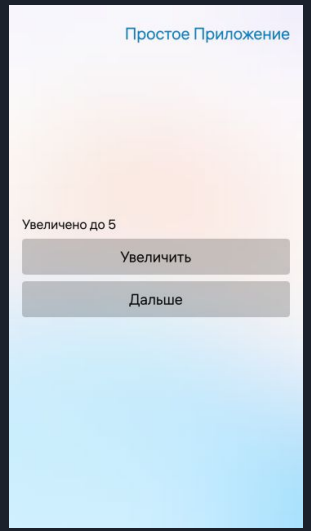
```
class MainVM : public QObject
{
    Q_PROPERTY(int counter READ counter WRITE setCounter NOTIFY counterChanged)
    int m_counter = 0;
public:
    // ...
    int counter() const { return m_counter; }
    void setCounter(int counter) {
        m_counter = counter;
        emit counterChanged(m_counter);
    }
    // ...
signals:
    void counterChanged(int counter);
    // ...
public slots:
    // ...
};
```



C++

mainvm.h

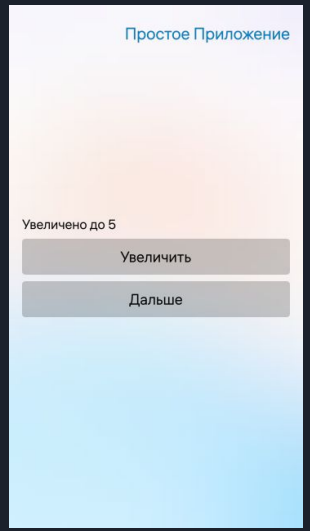
```
class MainVM : public QObject
{
    Q_PROPERTY(int counter READ counter WRITE setCounter NOTIFY counterChanged)
    int m_counter = 0;
public:
    // ...
    int counter() const { return m_counter; }
    void setCounter(int counter) {
        m_counter = counter;
        emit counterChanged(m_counter);
    }
    // ...
signals:
    void counterChanged(int counter);
    // ...
public slots:
    // ...
};
```



C++

mainvm.h

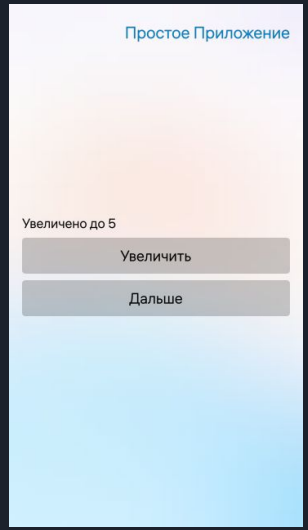
```
class MainVM : public QObject
{
    Q_PROPERTY(int counter READ counter WRITE setCounter NOTIFY counterChanged)
    int m_counter = 0;
public:
    // ...
    int counter() const { return m_counter; }
    void setCounter(int counter) {
        m_counter = counter;
        emit counterChanged(m_counter);
    }
    // ...
signals:
    void counterChanged(int counter);
    // ...
public slots:
    // ...
};
```



mainvm.h

```
class MainVM : public QObject
{
    // ...
public:
    // ...
    int counter() const { return m_counter; }
    void setCounter(int counter) { ... }
    Q_INVOKABLE void increase() { setCounter(m_counter + 1); }
    Q_INVOKABLE void next() { emit nextPressed(m_counter); }

signals:
    void counterChanged(int counter);
    void nextPressed(int);

public slots:
    void decreased(int counter) { setCounter(counter); }
};
```

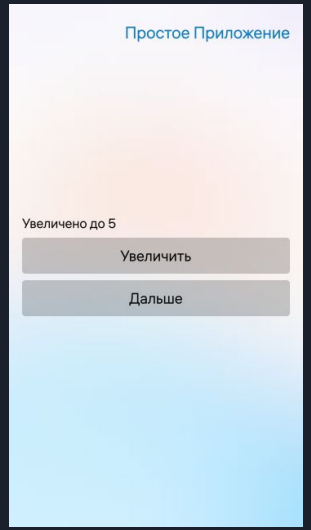
C++



mainvm.h

```
class MainVM : public QObject
{
    // ...
public:
    // ...
    int counter() const { return m_counter; }
    void setCounter(int counter) { ... }
    Q_INVOKABLE void increase() { setCounter(m_counter + 1); }
    Q_INVOKABLE void next() { emit nextPressed(m_counter); }

signals:
    void counterChanged(int counter);
    void nextPressed(int);

public slots:
    void decreased(int counter) { setCounter(counter); }
};
```

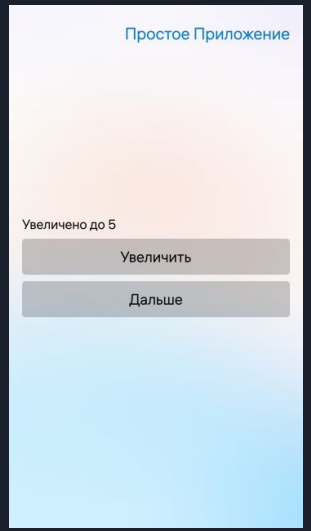
C++



C++

mainvm.h

```
class MainVM : public QObject
{
    // ...
public:
    // ...
    int counter() const { return m_counter; }
    void setCounter(int counter) { ... }
    Q_INVOKABLE void increase() { setCounter(m_counter + 1); }
    Q_INVOKABLE void next() { emit nextPressed(m_counter); }

signals:
    void counterChanged(int counter);
    void nextPressed(int);

public slots:
    void decreased(int counter) { setCounter(counter); }
};
```



C++

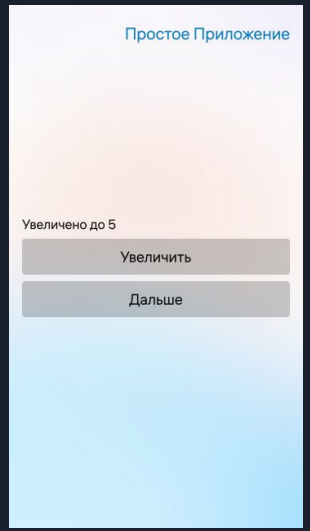
main.cpp

```cpp
#include "mainvm.h"

int main(int argc, char *argv[])
{
    qmlRegisterType<MainVM>("CustomCppClasses.Module", 1, 0, "MainVM");

    QScopedPointer<QGuiApplication> application(Aurora::Application::application(argc, argv));
    application->setOrganizationName(QStringLiteral("com.den3000"));
    application->setApplicationName(QStringLiteral("BasicApp"));

    QScopedPointer<QQuickView> view(Aurora::Application::createView());
    view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/TrivialApp.qml")));
    view->show();

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

QML

MainPage.qml

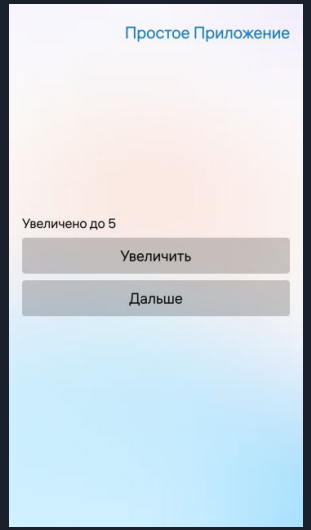
```
Page {
    MainVM { id: vm }
    property MainVM viewModel: vm
    // ...
    Column {
        // ...
        Text {
            // ...
            text: qsTr("Увеличено до %1").arg(viewModel.counter)
        }
        Button {
            // ...
            onClicked: viewModel.increase()
        }
        Button {
            // ...
            onClicked: viewModel.next()
        }
```



MainPage.qml

```
Page {
    MainVM { id: vm }
    property MainVM viewModel: vm
    // ...
    Column {
        // ...
        Text {
            // ...
            text: qsTr("Увеличено до %1").arg(viewModel.counter)
        }
        Button {
            // ...
            onClicked: viewModel.increase()
        }
        Button {
            // ...
            onClicked: viewModel.next()
        }
```

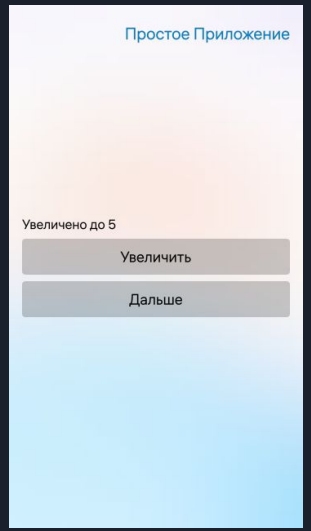
QML



MainPage.qml

```
Page {
    MainVM { id: vm }
    property MainVM viewModel: vm
    // ...
    Column {
        // ...
        Text {
            // ...
            text: qsTr("Увеличено до %1").arg(viewModel.counter)
        }
        Button {
            // ...
            onClicked: viewModel.increase()
        }
        Button {
            // ...
            onClicked: viewModel.next()
        }
```

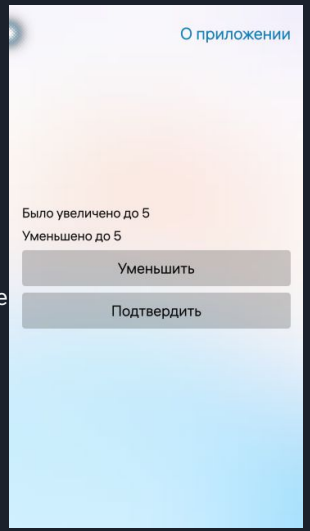
QML



C++

aboutvm.h

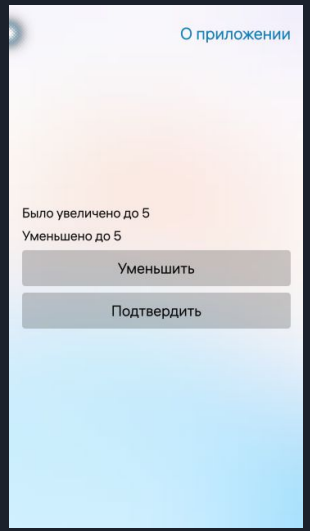
```
class AboutVM : public QObject
{
    Q_PROPERTY(int inputCounter READ inputCounter WRITE setInputCounter NOTIFY inputCounterChange
    Q_PROPERTY(int counter READ counter WRITE setCounter NOTIFY counterChanged)
    int m_inputCounter = 0;
    int m_counter = 0;

public:
    int inputCounter() const { return m_inputCounter; }
    void setInputCounter(int counter) { ... }
    int counter() const { return m_counter; }
    void setCounter(int counter) { ... }
    Q_INVOKABLE void decrease() { ... }
    Q_INVOKABLE void confirm() { ... }

// ...
};
```



C++

aboutvm.h

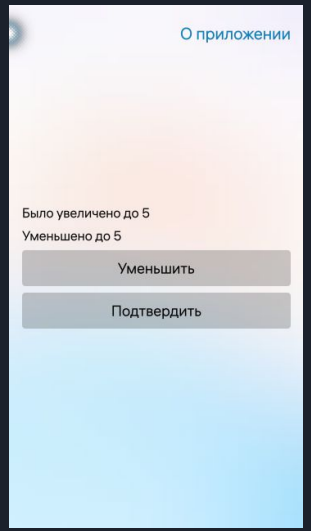
```
class AboutVM : public QObject
{
    // ...
public:
    int inputCounter() const { return m_inputCounter; }
    void setInputCounter(int counter) { ... }
    int counter() const { return m_counter; }
    void setCounter(int counter) { ... }
    Q_INVOKABLE void decrease() { ... }
    Q_INVOKABLE void confirm() { ... }

signals:
    void counterChanged(int counter);
    void inputCounterChanged(int counter);
    void confirmPressed(int counter);
};
```



QML

AboutPage.qml

```
Page {
    AboutVM { id: vm }
    property AboutVM viewModel: vm
    Column {
        Text {
            text: qsTr("Было увеличено до %1").arg(viewModel.inputCounter)
        }
        Text {
            text: qsTr("Counter: %1").arg(viewModel.counter)
        }
        Button {
            onClicked: viewModel.decrease()
        }
        Button {
            onClicked: viewModel.confirm()
        }
    }
}
```



QML

Coordinator.qml

```
Item {
    signal onDecreaseConfirmed(int counter)

    function start() {
        var page = pushPage(mainPage)
        page.viewModel.nextPressed.connect(showAbout)
        onDecreaseConfirmed.connect(page.viewModel.decreased)
    }

    function showAbout(counter) {
        var page = pushPage(aboutPage)
        page.viewModel.inputCounter = counter
        page.viewModel.counter = counter
        page.viewModel.confirmPressed.connect(onDecreaseConfirmed)
        page.viewModel.confirmPressed.connect(pop)
    }
}
```

QML

Coordinator.qml

```
Item {
    signal onDecreaseConfirmed(int counter)

    function start() {
        var page = pushPage(mainPage)
        page.viewModel.nextPressed.connect(showAbout)
        onDecreaseConfirmed.connect(page.viewModel.decreased)
    }

    function showAbout(counter) {
        var page = pushPage(aboutPage)
        page.viewModel.inputCounter = counter
        page.viewModel.counter = counter
        page.viewModel.confirmPressed.connect(onDecreaseConfirmed)
        page.viewModel.confirmPressed.connect(pop)
    }
}
```

QML

Coordinator.qml

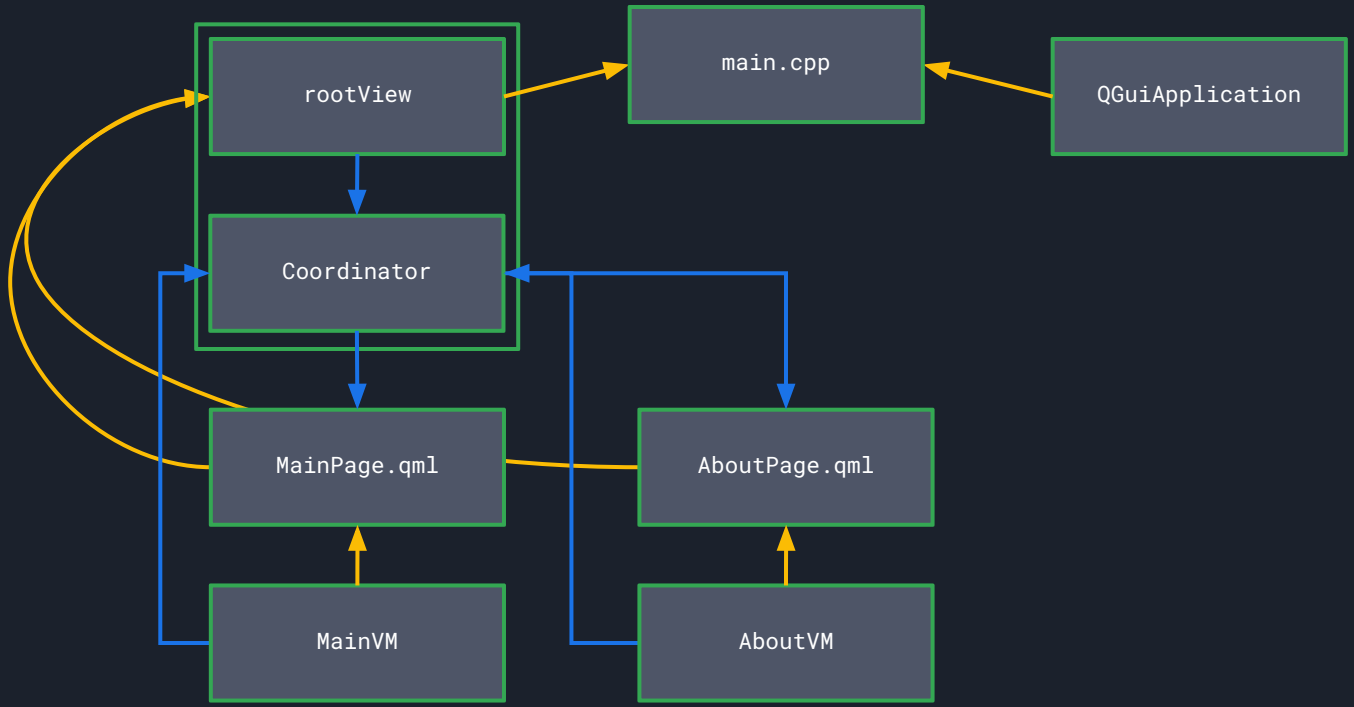
```
Item {
    signal onDecreaseConfirmed(int counter)

    function start() {
        var page = pushPage(mainPage)
        page.viewModel.nextPressed.connect(showAbout)
        onDecreaseConfirmed.connect(page.viewModel.decreased)
    }

    function showAbout(counter) {
        var page = pushPage(aboutPage)
        page.viewModel.inputCounter = counter
        page.viewModel.counter = counter
        page.viewModel.confirmPressed.connect(onDecreaseConfirmed)
        page.viewModel.confirmPressed.connect(pop)
    }
}
```

QML

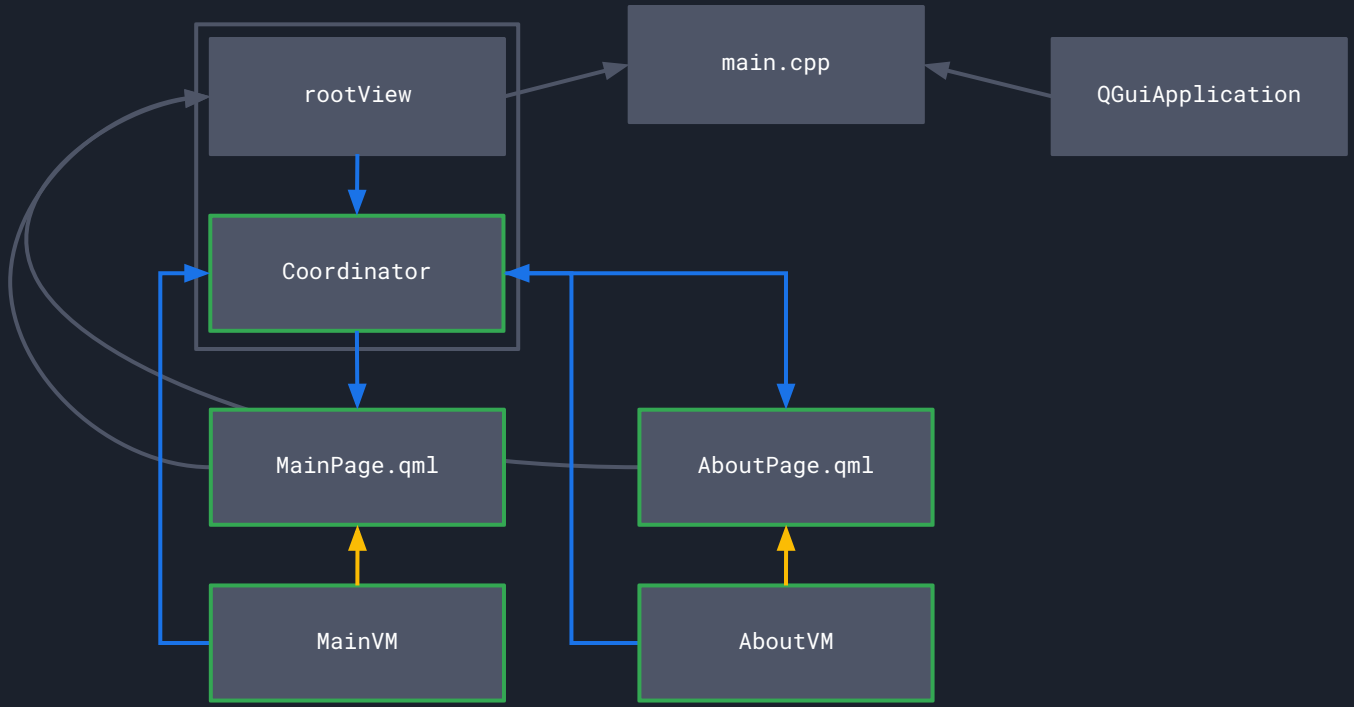
Coordinator.qml

```
Item {
    signal onDecreaseConfirmed(int counter)

    function start() {
        var page = pushPage(mainPage)
        page.viewModel.nextPressed.connect(showAbout)
        onDecreaseConfirmed.connect(page.viewModel.decreased)
    }

    function showAbout(counter) {
        var page = pushPage(aboutPage)
        page.viewModel.inputCounter = counter
        page.viewModel.counter = counter
        page.viewModel.confirmPressed.connect(onDecreaseConfirmed)
        page.viewModel.confirmPressed.connect(pop)
    }
}
```

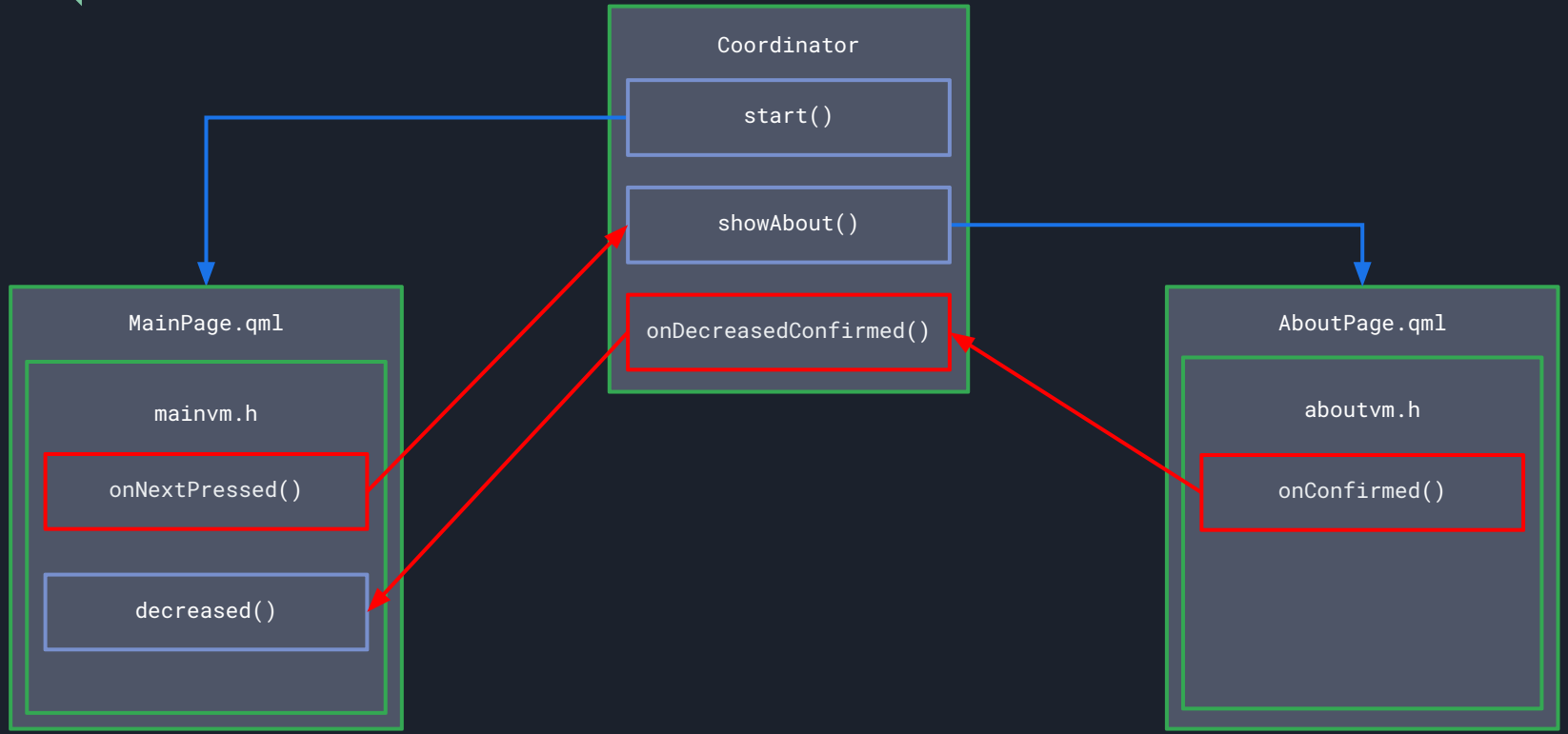
App Flow


rootView
main.cpp
QGuiApplication
Coordinator
MainPage.qml
AboutPage.qml
MainVM
AboutVM

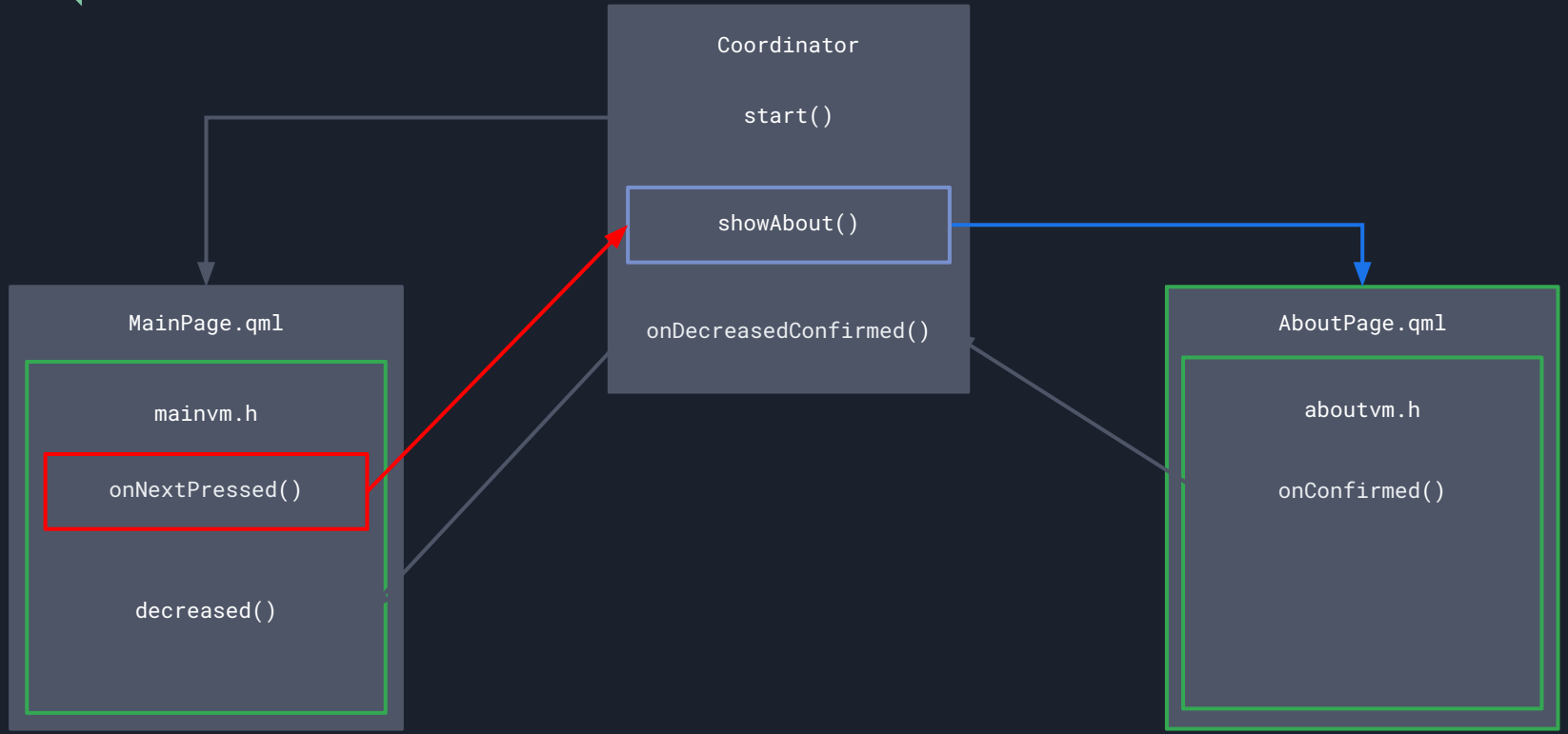
App Flow


rootView
main.cpp
QGuiApplication
Coordinator
MainPage.qml
AboutPage.qml
MainVM
AboutVM

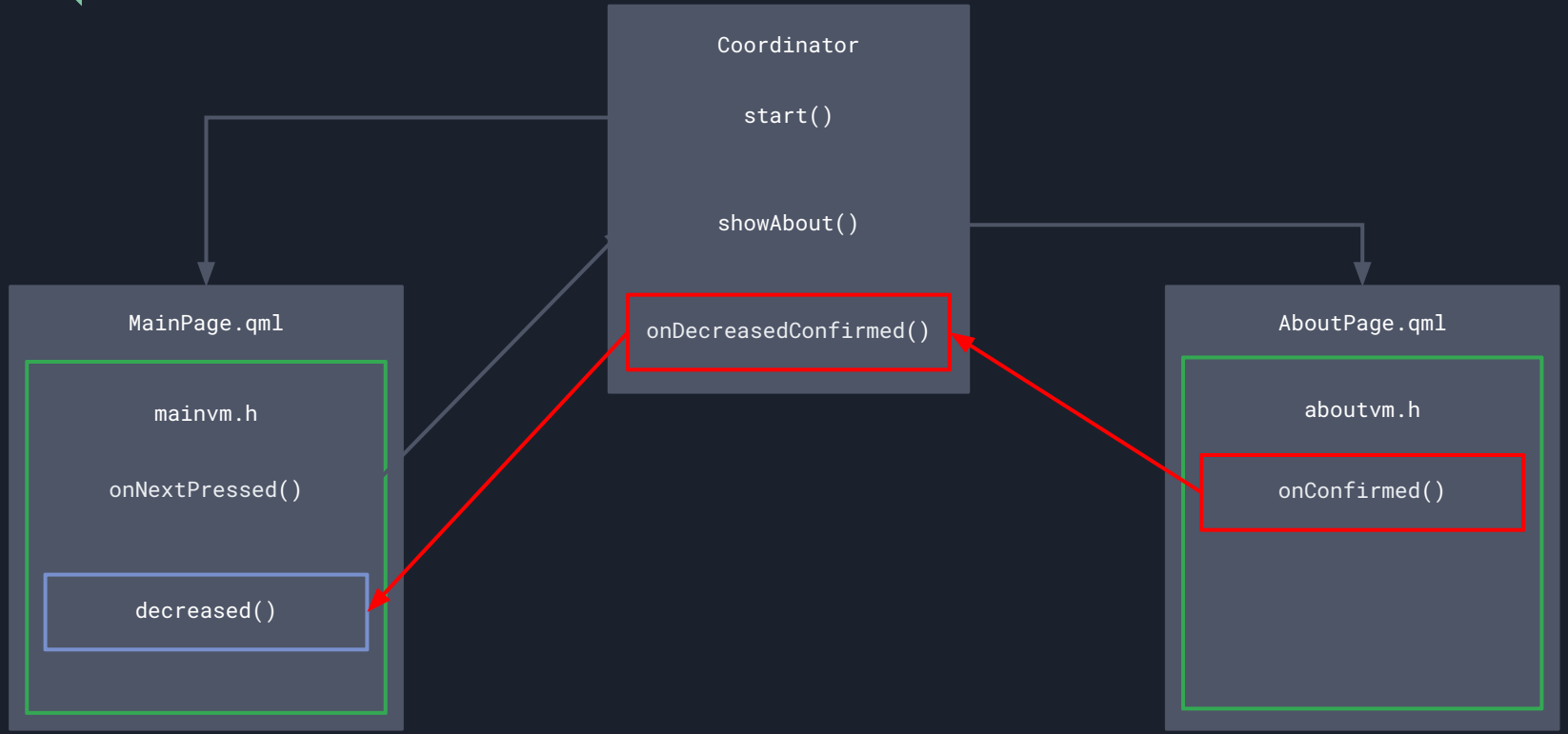
Coordinator.qml - Signals and Slots



Coordinator.qml - Signals and Slots



Coordinator.qml - Signals and Slots



Проблемы простого приложения

- ~~Навигация “вшита” в экраны~~
- ~~MainPage “знает” слишком много об AboutPage~~
- ~~Нет возможности динамически выбирать стартовую страницу~~
- ~~Нет разделения UI и логики~~
- Coordinator ответственен за инстанцирование ViewModel
- ViewModel инстанцируется на стороне QML, и мы не можем заинджектить в неё обычный C++ сервис

Дорожная карта

1. Аврора ОС
2. Инструменты разработчика
3. CLEAN архитектура и Coordinator
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. DI
8. Итоги

Дорожная карта

1. Аврора ОС
2. Инструменты разработчика
3. CLEAN архитектура и Coordinator
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. Бонус: C++
8. Итоги


У тебя есть ARC?
Лучше
У меня есть QObject::parent и умные указатели

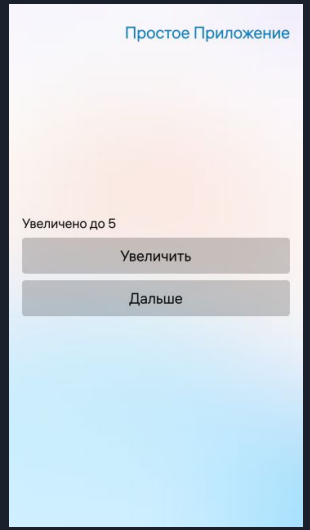
QObject::parent

1. Если у QObject есть родитель, то когда родитель будет удалён из памяти, этот дочерний объект тоже будет удалён из памяти, как и все остальные дочерние объекты этого родителя
2. Таким образом строится дерево объектов, которые все вместе удаляются из памяти при удалении родителя

mainvm.h

```
class MainVM : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QObject * parent READ parent WRITE setParent) // !!! IMPORTANT !!!

public:
    explicit MainVM(QObject *parent = nullptr): QObject(parent) { qDebug(); };
    ~MainVM() { qDebug(); }

    Q_INVOKABLE void nextPressed() { qDebug(); };
};
```

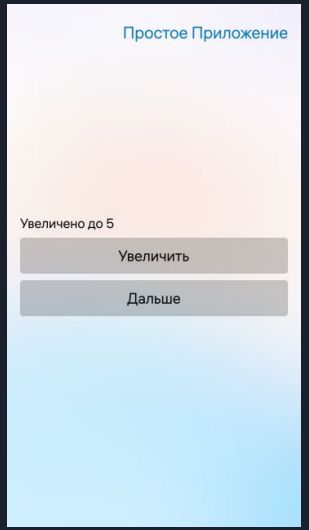
C++



MainPage.qml

```
Page {
    // ...

    MainVM {
        id: viewModel
    }

    // ...
}
```

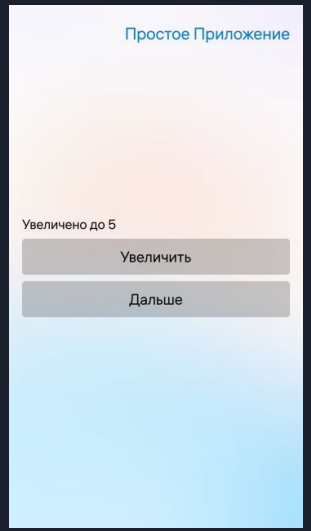
QML



MainPage.qml

```
Page {
    // ...

    property MainVM viewModel
    onViewModelChanged: viewModel.parent = this


    // ...
}
```

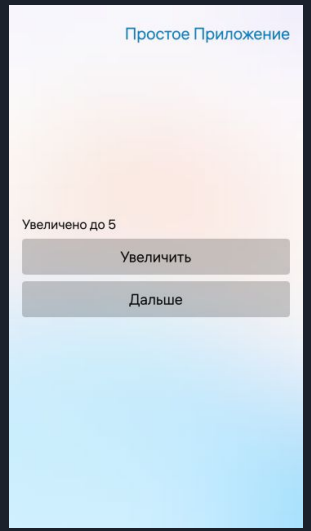
QML



MainPage.qml

```
Page {
    // ...

    property MainVM viewModel
    onViewModelChanged: viewModel.parent = this


    // ...
}
```

QML



Умные указатели

1. **shared_ptr** - по сути ARC
2. **weak_ptr** - для разрыва циклических зависимостей при использовании ARC, как weak в swift
3. **unique_ptr** - удобная штука для выражения владения объектом и передачи ответственности за него

C++

Умные указатели

```
auto service = make_shared<Service>(/*arguments*/);

auto someHeaveObject = make_unique<HeavyObject>(/*arguments*/);
...
// end of function

auto someServiceInstance() {
    return make_unique<StartService>(restApi());
}

auto someService = someServiceInstance();  // someService is unique_ptr
someService->                              // someService fields and functions
someService.                               // unique_ptr fields and functions

shared_ptr someService = someServiceInstance(); // someService is shared_ptr
someService->                              // someService fields and functions
someService.                               // shared_ptr fields and functions
```

C++

Умные указатели

```
auto service = make_shared<Service>(/*arguments*/);

auto someHeaveObject = make_unique<HeavyObject>(/*arguments*/);
...
// end of function

auto someServiceInstance() {
    return make_unique<StartService>(restApi());
}

auto someService = someServiceInstance();  // someService is unique_ptr
someService->                              // someService fields and functions
someService.                               // unique_ptr fields and functions

shared_ptr someService = someServiceInstance(); // someService is shared_ptr
someService->                              // someService fields and functions
someService.                               // shared_ptr fields and functions
```

C++

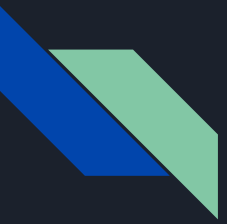
Умные указатели

```
auto service = make_shared<Service>(/*arguments*/);

auto someHeaveObject = make_unique<HeavyObject>(/*arguments*/);
...
// end of function

auto someServiceInstance() {
    return make_unique<StartService>(restApi());
}

auto someService = someServiceInstance();  // someService is unique_ptr
someService->                              // someService fields and functions
someService.                               // unique_ptr fields and functions

shared_ptr someService = someServiceInstance(); // someService is shared_ptr
someService->                              // someService fields and functions
someService.                               // shared_ptr fields and functions
```

C++

Умные указатели

```
auto service = make_shared<Service>(/*arguments*/);

auto someHeaveObject = make_unique<HeavyObject>(/*arguments*/);
...
// end of function

auto someServiceInstance() {
    return make_unique<StartService>(restApi());
}

auto someService = someServiceInstance();  // someService is unique_ptr
someService->                              // someService fields and functions
someService.                               // unique_ptr fields and functions

shared_ptr someService = someServiceInstance(); // someService is shared_ptr
someService->                              // someService fields and functions
someService.                               // shared_ptr fields and functions
```



.pro

```
CONFIG += \
    auroraapp\
    c++1z
```

C++

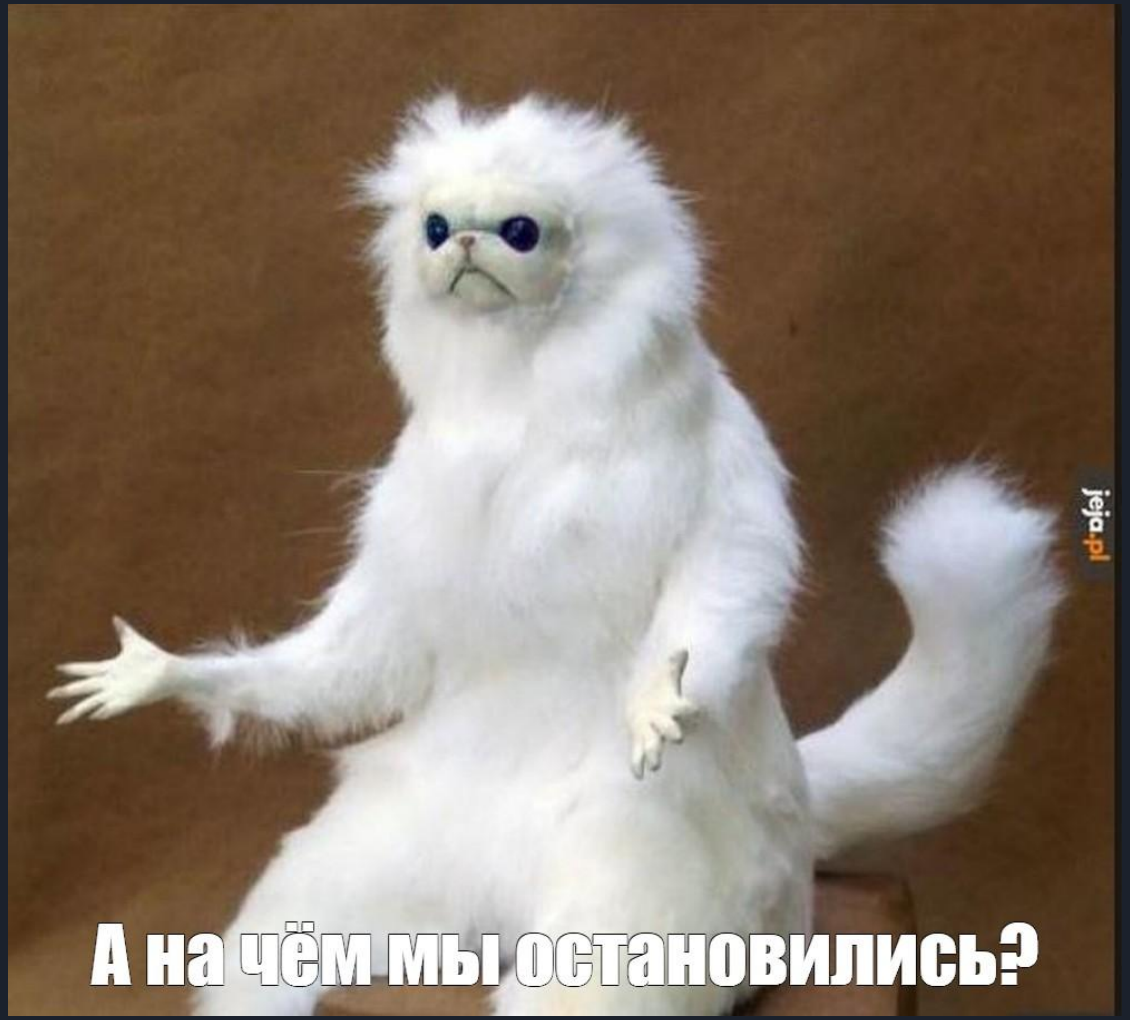
easy_import.h

```
#include <memory>

using std::shared_ptr;
using std::make_shared;
using std::weak_ptr;
using std::unique_ptr;
using std::make_unique;

template <typename R>
constexpr inline R * unique_unwrap(unique_ptr<R> && unq) {
    R * result = unq.get();
    unq.release();
    return result;
}
```

C++

easy_import.h

```
#include <memory>

using std::shared_ptr;
using std::make_shared;
using std::weak_ptr;
using std::unique_ptr;
using std::make_unique;

template <typename R>
constexpr inline R * unique_unwrap(unique_ptr<R> && unq) {
    R * result = unq.get();
    unq.release();
    return result;
}
```


jeja.pl
А на чём мы остановились?

Дорожная карта

1. Аврора ОС
2. Инструменты разработчика
3. Архитектура - её важность и эволюция в мобильной разработке
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. DI
8. Итоги

C++

mathservice.h

```
class MathService {
    int m_increment;
public:
    explicit MathService(int increment)
        : m_increment { increment }
    { qDebug(); };
    ~MathService() { qDebug(); }

    int increaseValue(int value) { return value + m_increment; }

    int decreaseValue(int value) { return value - m_increment; }
};
```

C++

mathservice.h

```
class MathService {
    int m_increment;
public:
    explicit MathService(int increment)
        : m_increment { increment }
    { qDebug(); };
    ~MathService() { qDebug(); }

    int increaseValue(int value) { return value + m_increment; }

    int decreaseValue(int value) { return value - m_increment; }
};
```

C++

dicontainer.h

```
class DiContainer {
public:
    unique_ptr<MathService> mathServiceInstance()
        { return make_unique<MathService>(increment()); }

    unique_ptr<MainVM> mainVmInstance(shared_ptr<MathService> service)
        { return make_unique<MainVM>(service); }

    unique_ptr<AboutVM> aboutVmInstance(int counter, shared_ptr<MathService> service)
        { return make_unique<AboutVM>(counter, service); }
protected:
    virtual int increment() = 0;
};
```

C++

dicontainer.h

```
class DiContainer {
public:
    unique_ptr<MathService> mathServiceInstance()
        { return make_unique<MathService>(increment()); }

    unique_ptr<MainVM> mainVmInstance(shared_ptr<MathService> service)
        { return make_unique<MainVM>(service); }

    unique_ptr<AboutVM> aboutVmInstance(int counter, shared_ptr<MathService> service)
        { return make_unique<AboutVM>(counter, service); }
protected:
    virtual int increment() = 0;
};
```

C++

dicontainer.h

```
class DiContainer {
public:
    unique_ptr<MathService> mathServiceInstance()
        { return make_unique<MathService>(increment()); }

    unique_ptr<MainVM> mainVmInstance(shared_ptr<MathService> service)
        { return make_unique<MainVM>(service); }

    unique_ptr<AboutVM> aboutVmInstance(int counter, shared_ptr<MathService> service)
        { return make_unique<AboutVM>(counter, service); }
protected:
    virtual int increment() = 0;
};
```

C++

diprovider.h

```
class MyDiContainer: public DiContainer {
protected:
    int increment() { return 1; }
};

class DiProvider: public QObject {
    Q_OBJECT
    MyDiContainer diContainer;
    // ...
public:
    explicit DiProvider(QObject * parent = nullptr): QObject(parent){ qDebug(); }
    ~DiProvider() { qDebug(); }
    // ...
};
```

C++

diprovider.h

```
class MyDiContainer: public DiContainer {
protected:
    int increment() { return 1; }
};

class DiProvider: public QObject {
    Q_OBJECT
    MyDiContainer diContainer;
    // ...
public:
    explicit DiProvider(QObject * parent = nullptr): QObject(parent){ qDebug(); }
    ~DiProvider() { qDebug(); }
    // ...
};
```

C++

diprovider.h

```
class MyDiContainer: public DiContainer {
protected:
    int increment() { return 1; }
};

class DiProvider: public QObject {
    Q_OBJECT
    MyDiContainer diContainer;
    // ...
public:
    explicit DiProvider(QObject * parent = nullptr): QObject(parent){ qDebug(); }
    ~DiProvider() { qDebug(); }
    // ...
};
```

C++

diprovider.h

```
class DiProvider: public QObject {
    Q_OBJECT

    MyDiContainer diContainer;

    shared_ptr<MathService> m_mathService;
    shared_ptr<MathService> lazyMathService() {
       if(!m_mathService) {
           m_mathService = diContainer.mathServiceInstance();
       }
       return m_mathService;
    };
public:
    // ...
};
```

C++

diprovider.h

```
class DiProvider: public QObject {
    shared_ptr<MathService> lazyMathService() { ... };

public:
    // ...

    Q_INVOKABLE MainVM * mainVmInstance()
        { return unique_unwrap(diContainer.mainVmInstance(lazyMathService())); }

    Q_INVOKABLE AboutVM * aboutVmInstance(int counter)
        { return unique_unwrap(diContainer.aboutVmInstance(counter, lazyMathService())); }
};
```

C++

diprovider.h

```
class DiProvider: public QObject {
    shared_ptr<MathService> lazyMathService() { ... };

public:
    // ...

    Q_INVOKABLE MainVM * mainVmInstance()
        { return unique_unwrap(diContainer.mainVmInstance(lazyMathService())); }

    Q_INVOKABLE AboutVM * aboutVmInstance(int counter)
        { return unique_unwrap(diContainer.aboutVmInstance(counter, lazyMathService())); }
};
```

C++

main.cpp

```cpp
#include "diprovider.h"

int main(int argc, char *argv[])
{
    // ...
    qmlRegisterType<DiProvider>("CustomCppClasses.Module", 1, 0, "DiProvider");
    QScopedPointer<QGuiApplication> application(Aurora::Application::application(argc, argv));
    // ...
    auto diProvider = make_shared<DiProvider>();
    QScopedPointer<QQuickView> view(Aurora::Application::createView());
    view->rootContext()->setContextProperty("diProvider", diProvider.get());
    view->setSource(Aurora::Application::pathTo(QStringLiteral("qml/BasicApp.qml")));
    view->show();
    return application->exec();
}
```

QML

Coordinator.qml

```
Item {
    function pushPageWithVm(path, vm) {
        return pageStack.push(
            Qt.createComponent(Qt.resolvedUrl(path)),
            { "viewModel" : vm}
        )
    }

    function pop() { pageStack.pop() }

    function start() { ... }

    function showAbout(counter) { ... }
}
```

QML

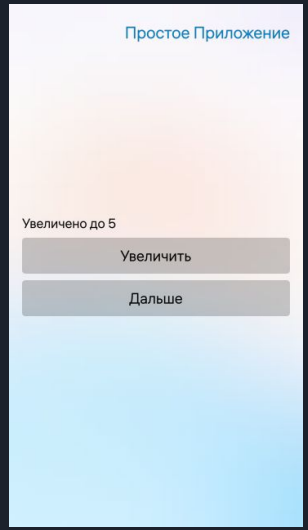
Coordinator.qml

```
Item {
    function pushPageWithVm(path, vm) { ... }

    function start() {
        var vm = diProvider.mainVmInstance()
        vm.nextPressed.connect(showAbout)
        onDecreaseConfirmed.connect(vm.decreased)
        pushPageWithVm(mainPage, vm)
    }

    function showAbout(counter) {
        var vm = diProvider.aboutVmInstance(counter)
        vm.confirmPressed.connect(onDecreaseConfirmed)
        vm.confirmPressed.connect(pop)
        pushPageWithVm(aboutPage, vm)
    }
}
```

MainPage.qml

```
Page {
    property MainVM viewModel
    onViewModelChanged: viewModel.parent = this

    PageHeader { title: qsTr("Простое Приложение") }

    Column {
        // ...

    }

}
```

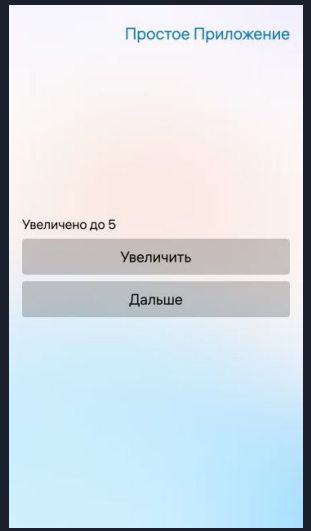
QML



C++

mainvm.h

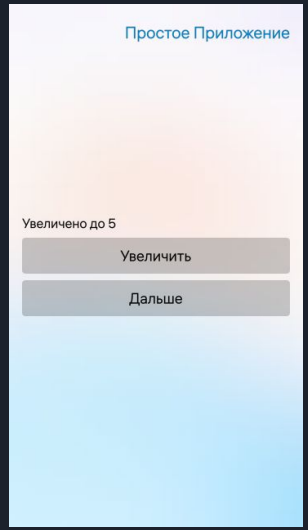
```
class MainVM : public QObject
{
    Q_PROPERTY(QObject * parent READ parent WRITE setParent) // !!! IMPORTANT !!!

    shared_ptr<MathService> m_service;
public:
    explicit MainVM(shared_ptr<MathService> service, QObject *parent = nullptr)
        : QObject(parent)
        , m_service { service }
    { qDebug(); };
    ~MainVM() { qDebug(); }
    Q_INVOKABLE void increase() { setCounter(m_service->increaseValue(m_counter)); }
signals:
public slots:
};
```



C++

mainvm.h

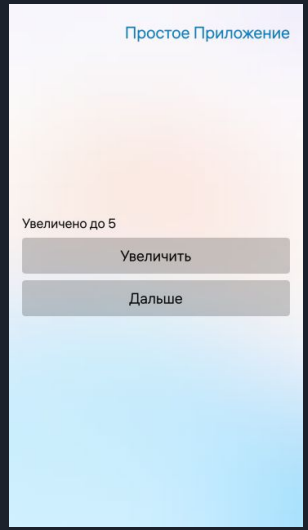
```
class MainVM : public QObject
{
    Q_PROPERTY(QObject * parent READ parent WRITE setParent) // !!! IMPORTANT !!!

    shared_ptr<MathService> m_service;
public:
    explicit MainVM(shared_ptr<MathService> service, QObject *parent = nullptr)
        : QObject(parent)
        , m_service { service }
    { qDebug(); };
    ~MainVM() { qDebug(); }
    Q_INVOKABLE void increase() { setCounter(m_service->increaseValue(m_counter)); }
signals:
public slots:
};
```



C++

mainvm.h

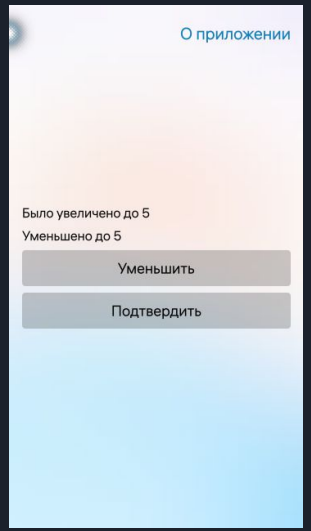
```
class MainVM : public QObject
{
    Q_PROPERTY(QObject * parent READ parent WRITE setParent) // !!! IMPORTANT !!!

    shared_ptr<MathService> m_service;
public:
    explicit MainVM(shared_ptr<MathService> service, QObject *parent = nullptr)
        : QObject(parent)
        , m_service { service }
    { qDebug(); };
    ~MainVM() { qDebug(); }
    Q_INVOKABLE void increase() { setCounter(m_service->increaseValue(m_counter)); }
signals:
public slots:
};
```



QML

AboutPage.qml

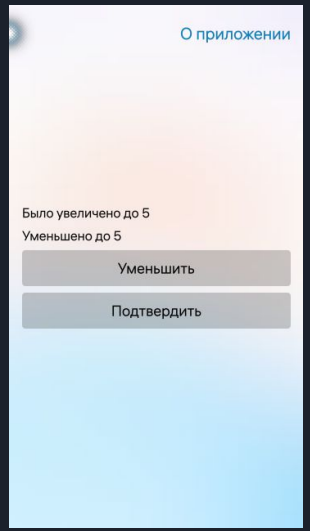
```
Page {
    property AboutVM viewModel
    onViewModelChanged: viewModel.parent = this

    PageHeader { title: qsTr("Простое Приложение") }

    Column {
        // ...

    }

}
```



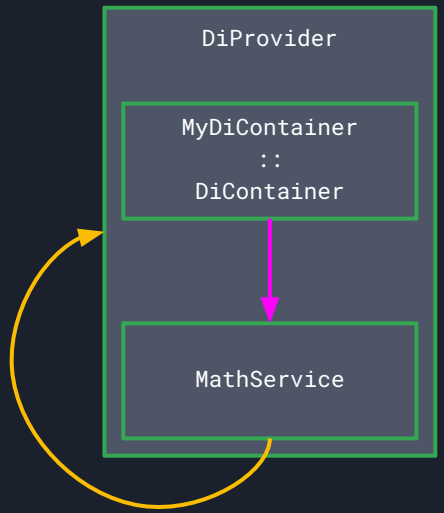
C++

aboutvm.h

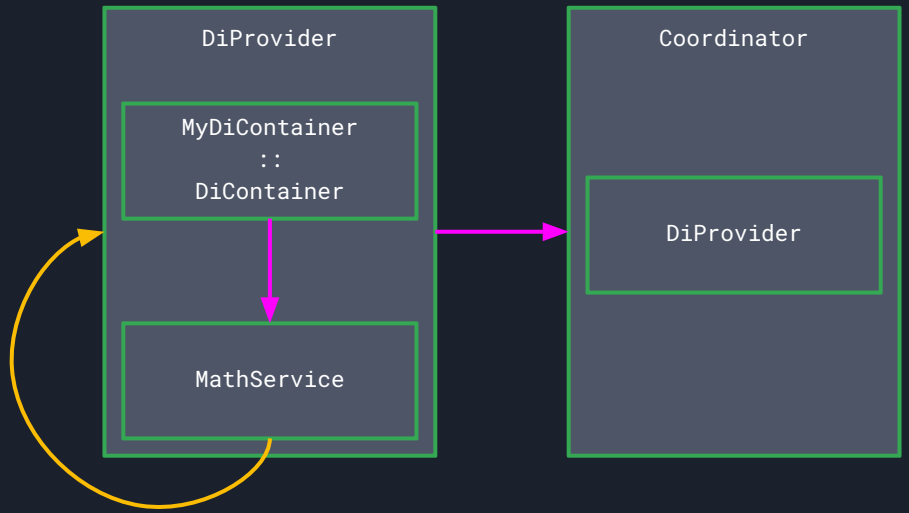
```
class AboutVM: public QObject
{
    Q_PROPERTY(QObject * parent READ parent WRITE setParent) // !!! IMPORTANT !!!

    shared_ptr<MathService> m_service;
public:
    explicit AboutVM(shared_ptr<MathService> service, QObject *parent = nullptr)
        : QObject(parent)
        , m_service { service }
    { qDebug(); };
    ~AboutVM() { qDebug(); }

signals:
public slots:
};
```



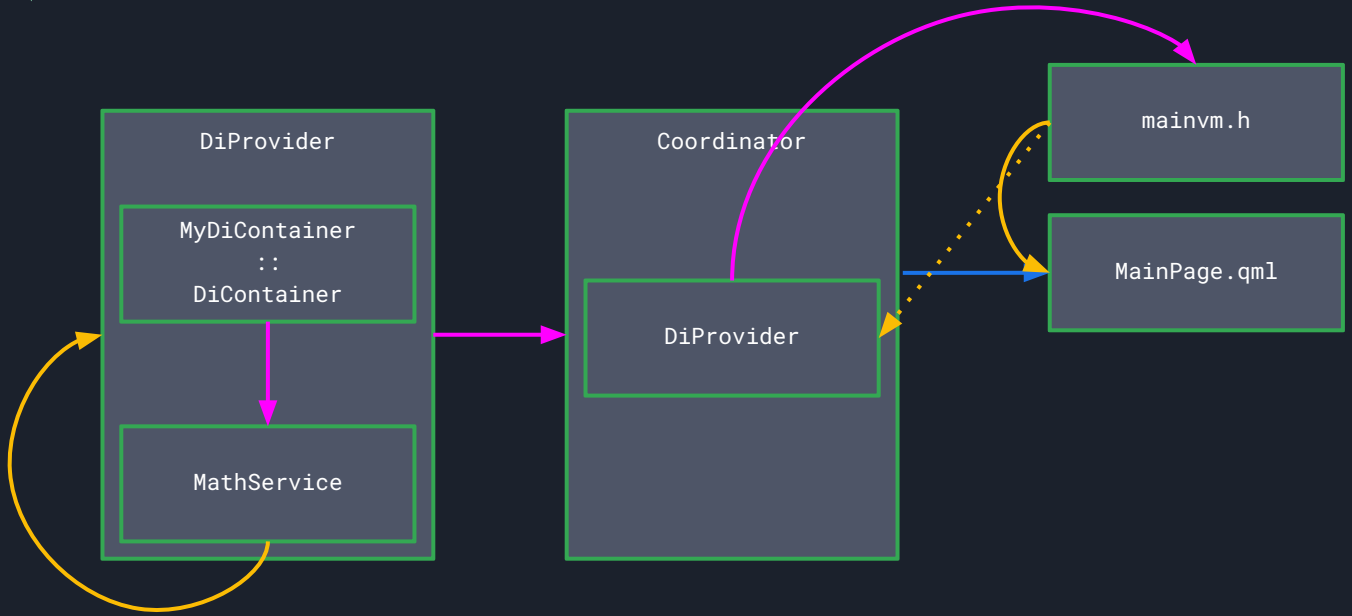
DI Flow



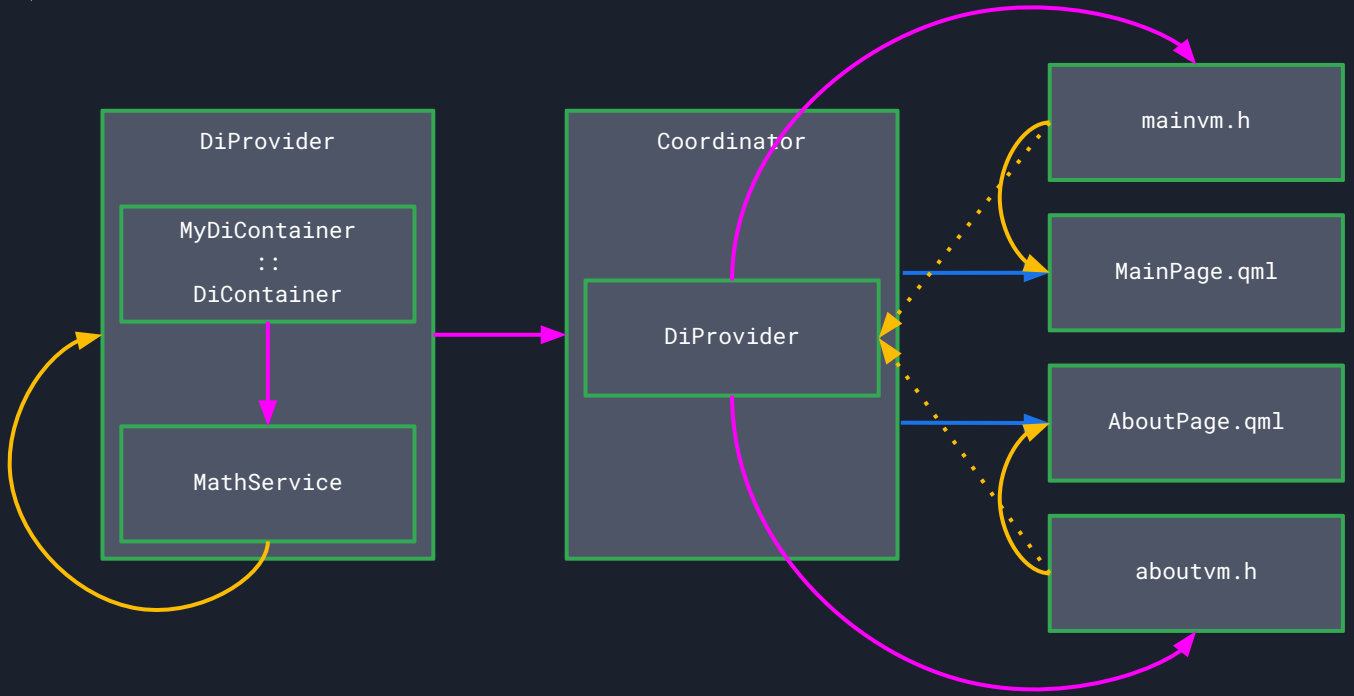
DI Flow


DiProvider
MyDiContainer
::
DiContainer
MathService
Coordinator
DiProvider

DI Flow


DiProvider
MyDiContainer
::
DiContainer
MathService
Coordinator
DiProvider
mainvm.h
MainPage.qml

DI Flow


DiProvider
MyDiContainer
::
DiContainer
MathService
Coordinator
DiProvider
mainvm.h
MainPage.qml
AboutPage.qml
aboutvm.h

Проблемы простого приложения

- ~~Навигация “вшита” в экраны~~
- ~~MainPage “знает” слишком много об AboutPage~~
- ~~Нет возможности динамически выбирать стартовую страницу~~
- ~~Нет разделения UI и логики~~
- ~~Coordinator ответственен за инстанцирование ViewModel~~
- ~~ViewModel инстанцируется на стороне QML, и мы не можем заинджектить в неё обычный C++ сервис~~

Дорожная карта

1. Аврора ОС
2. Инструменты разработчика
3. Архитектура - её важность и эволюция в мобильной разработке
4. Пример простого приложения из двух экранов
5. Coordinator
6. ViewModel
7. DI
8. Итоги

Выводы

1. При разработке для Аврора ОС у нас есть весь необходимый набор инструментов, чтобы строить приложения с применением известных “лучших практик”
2. QML даёт нам возможность строить приложением привычным декларативным способом, в том числе и навигацию
3. C++ в 2023 году это совсем не больно, хотя и требует некоторой адаптации


Теперь я знаю как разрабатывать
под Аврора ОС

Спасибо за внимание

1. Скачать Aurora IDE
   https://developer.auroraos.ru/#tree
2. Исходный код BasicApp-Aurora
   https://github.com/den3000/BasicApp-Aurora
3. Исходный код BasicApp-SwiftUI
   https://github.com/den3000/BasicApp-SwiftUI
4. Исходный код ToDoFeed-Aurora
   https://github.com/den3000/ToDoFeed-aurora