МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра Математического и компьютерного моделирования

Организация UI в российской мобильной ОС «Аврора»

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студентки 2 курса 247 группы

направление 09.04.03 — Прикладная информатика

механико-математического факультета

Кондратовой Татьяны Сергеевны

Научный руководитель
зав. каф., д.ф.-м.н., доцент Ю.А. Блинков

Зав. кафедрой
зав. каф., д.ф.-м.н., доцент Ю.А. Блинков

Саратов 2023

Введение. ОС Аврора – первая отечественная операционная система для мобильных устройств. Ориентирована на крупный бизнес и государственные организации с большим штатом полевых сотрудников.

Разрабатывается на базе ядра свободного программного обеспечения Linux. Также в состав продукта входят несколько лицензированных закрытых компонент. Полный цикл производственной системы осуществляется в России российскими специалистами.

Аврора – единственная отечественная мобильная ОС, которая включена в Единый реестр российских программ для электронных вычислительных машин и баз данных. Сертифицирована ФСБ и ФСТЭК. У ОС есть собственная платформа управления устройствами – Аврора Центр.

Среди возможностей ОС Аврора можно выделить следующие:

- Доверенная загрузка и контроль целостности файловой системы
- Встроенная верификация установки и запуска программ
- Встроенные политики безопасности
- Полный дистанционный контроль над всеми функциями смартфона
- Собственная платформа управления устройствами
- Защита каналов связи (ГОСТ VPN)
- Многофакторная аутентификация (включая поддержку токенов)
- Шифрование данных
- Работа с электронной подписью (в том числе квалифицированной)
- ПО для решения любых задач: от прикладного (браузеры, мультимедиа, мессенджеры, и др.) до профессионального и специализированного(документооборот, управление персоналом, работа с доверенными файловыми хранилищами и др.).

«Открытая мобильная платформа» и УЦ «Основание» успешно протестировали работу электронной подписи на платформе «Аврора». Развитие отечественной операционной системы очень важно в условиях политики импортозамещения и цифрового суверенитета. Решение ОС Аврора с применением КЭП и сертификатов УЦ «Основание» позволит российским компаниям без ущерба для бизнес-процессов перейти на использование отечественного ПО.

Компания МойОфис сообщила о выпуске мобильного приложения «МойОфис Документы» для ОС Аврора. Решение позволяет работать с тексто-

выми и табличными документами, а также с презентациями на смартфонах и планшетах с защищенной операционной системой Аврора 4.0. Речь идет о первом офисном программном обеспечении для этой ОС.

Мобильное приложение «МойОфис Документы» для ОС Аврора предназначено для сотрудников государственных организаций и крупных коммерческих предприятий, которым требуется работа с документами в безопасной корпоративной среде и обеспечение полного контроля над данными. По словам разработчиков, решение обладает необходимыми функциями для осуществления типовой офисной деятельности вне рабочего места, например, на выездах и в командировках.

Первый раздел посвященн QML. QML – это декларативный язык, используемый для описания того, как объекты связаны друг с другом. QtQuick – это фреймворк, построенный на QML для создания пользовательского интерфейса вашего приложения. Он разбивает пользовательский интерфейс на более мелкие элементы, которые можно объединить в компоненты. QtQuick описывает внешний вид и поведение этих элементов пользовательского интерфейса. Это описание пользовательского интерфейса можно дополнить кодом JavaScript, чтобы обеспечить не только простую, но и более сложную логику. С этой точки зрения он следует шаблону HTML-JavaScript, но QML и QtQuick разработаны с нуля для описания пользовательских интерфейсов, а не текстовых документов.

В своей простейшей форме QtQuick позволяет создавать иерархию элементов. Дочерние элементы наследуют систему координат от родителя. Координаты x,y всегда выражается относительно родителя.

- Инструкция import импортирует модуль. Можно добавить необязательную версию в виде <мажорная версия>.<минорная версия>.
- Комментарии можно делать с помощью // для однострочных комментариев или /* */ для многострочных комментариев. Так же, как в C/C++ и JavaScript
- Каждый файл QML должен иметь ровно один корневой элемент, как в HTML.
- Элемент объявляется по его типу, за которым следует {}.

- Элементы могут иметь свойства, они представлены в форме имя:значение.
- Доступ к произвольным элементам внутри документа QML можно получить, используя их id (идентификатор без кавычек).
- Элементы могут быть вложенными, то есть родительский элемент может иметь дочерние элементы. Доступ к родительскому элементу можно получить с помощью ключевого слова parent

Пример QML документа:

```
import QtQuick 2.0
import Sailfish.Silica 1.0
Page {
  id: page
  allowedOrientations: Orientation.All
  SilicaFlickable {
    anchors.fill: parent
    contentHeight: column.height
    Column {
      id: column
      width: page.width
      spacing: Theme.paddingLarge
      PageHeader {
      title: qsTr("UI Template")
    }
    Label {
      x: Theme.horizontalPageMargin
      text: qsTr("Hello Sailors")
      color: Theme.secondaryHighlightColor
      font.pixelSize: Theme.fontSizeExtraLarge
    } }
} }
```

Элементы объявляются с использованием имени элемента, но определяются с помощью их свойств или путем создания пользовательских свойств. Свойство представляет собой простую пару ключ-значение, например. width: 100, text: 'Greetings', color: '#FF0000'. Свойство имеет четко определенный тип и может иметь начальное значение.

Рассмотрим различные особенности свойств:

1. id – это специальное значение, похожее на свойство, оно используется для ссылки на элементы внутри файла QML (называемого «документом»). id не является строковым типом, а является идентификатором и частью синтаксиса QML. id должен быть уникальным внутри документа, его нельзя сбросить на другое значение или запросить (он ведет себя так же, как ссылка в мире C++).
2. Свойству может быть присвоено значение в зависимости от его типа. Если для свойства не задано значение, будет выбрано начальное значение. Для получения дополнительной информации о начальном значении свойства вам необходимо обратиться к документации конкретного элемента.
3. Свойство может зависеть от одного или многих других свойств. Это называется привязкой. Связанное свойство обновляется при изменении свойств, от которых оно зависит. Это работает как контракт, в этом случае высота, height, всегда должна быть в два раза больше ширины, width.
4. Добавление новых свойств к элементу выполняется с использованием квалификатора property, за которым следует тип, имя и необязательное начальное значение (property <тип> <имя> : <значение>). Если начальное значение не задано, выбирается начальное значение по умолчанию.
5. Еще одним важным способом объявления свойств является использование ключевого слова alias (property alias <имя>: <ссылка>). Ключевое слово alias позволяет нам перенаправить свойство объекта или сам объект из типа во внешнюю область. Мы будем использовать эту технику позже при определении компонентов для экспорта внутренних свойств или идентификаторов элементов на корневой уровень. Псевдоним свойства не нуждается в типе, он использует тип свойства или объекта, на который ссылается.
6. Свойство text зависит от пользовательского свойства times типа int. Значение int автоматически преобразуется в тип string. Само выражение является еще одним примером привязки и приводит к обновлению текста при каждом изменении свойства times.

7. Некоторые свойства являются сгруппированными свойствами. Эта функция используется, когда свойство структурировано, и связанные свойства должны быть сгруппированы вместе. Другой способ записи сгруппированных свойств – это font family: "Ubuntu"; pixelSize: 24 .
8. Некоторые свойства принадлежат самому классу элементов. Это делается для элементов глобальных настроек, которые появляются в приложении только один раз (например, ввод с клавиатуры). Они записываются следующим образом: <Элемент>.<свойство>: <значение>.
9. Для каждого свойства можно указать обработчик сигнала. Этот обработчик вызывается после изменения свойства. Например, здесь мы хотим получать уведомления при изменении высоты и использовать встроенную консоль для логгирования сообщения в системе.

Элементы можно разделить на визуальные и невизуальные. Визуальный элемент (например, Rectangle, прямоугольник) имеет геометрическую форму и обычно представляет собой область на экране. Невизуальный элемент (например, Timer, таймер) обеспечивает общую функциональность, обычно используемую для управления визуальными элементами.

Для отображения текста можно использовать элемент Text. Наиболее заметным его свойством является свойство text типа string. Элемент вычисляет свою начальную ширину и высоту на основе заданного текста и используемого шрифта. На шрифт можно повлиять с помощью группы свойств шрифта (например, font.family, font.pixelSize, ...). Чтобы изменить цвет текста, просто используйте свойство color.

```
Text {
    text: "The quick brown fox"
    color: "#303030"
    font.family: "Ubuntu"
    font.pixelSize: 28
}
```

Во втором разделе рассмотрена витрина системных графических компонентов для ОС Аврора.

APPLICATIONWINDOW — точка входа в приложение:

- initialPage : var — определяет, какая страница будет загружена при запуске приложения
- cover : var — определяет обложку приложения
- pageStack : PageStack — стек отображаемых страниц,
- allowedOrientations : enumeration — набор доступных ориентаций экрана
  - Orientation.All
  - Orientation.PortraitMask
  - Orientation.Portrait
  - Orientation.PortraitInverted (Cover) не для смартфонов)
- Orientation.LandscapeMask
  - Orientation.Landscape
  - Orientation.LandscapeInverted
- activate() — вывести приложение в полноэкранный режим
- deactivate()— свернуть приложение и отобразить обложку на рабочем столе

Пример такого приложения в соответствии с рисунком 1:

```
import QtQuick 2.6
import Sailfish.Silica 1.0
ApplicationWindow {
  initialPage:
    Component { Page { } }
  cover:
    Component {
      CoverBackground { }
    }
  allowedOrientations: defaultAllowedOrientations
}
```

Обычная кнопка:

- preferredWidth : real — размер кнопки
  - Theme.buttonWidthExtraSmall
  - Theme.buttonWidthMedium
  - Theme.buttonWidthLarge
- text : string — отображаемый текст на кнопке

Пример такого приложения в соответствии с рисунком 2:

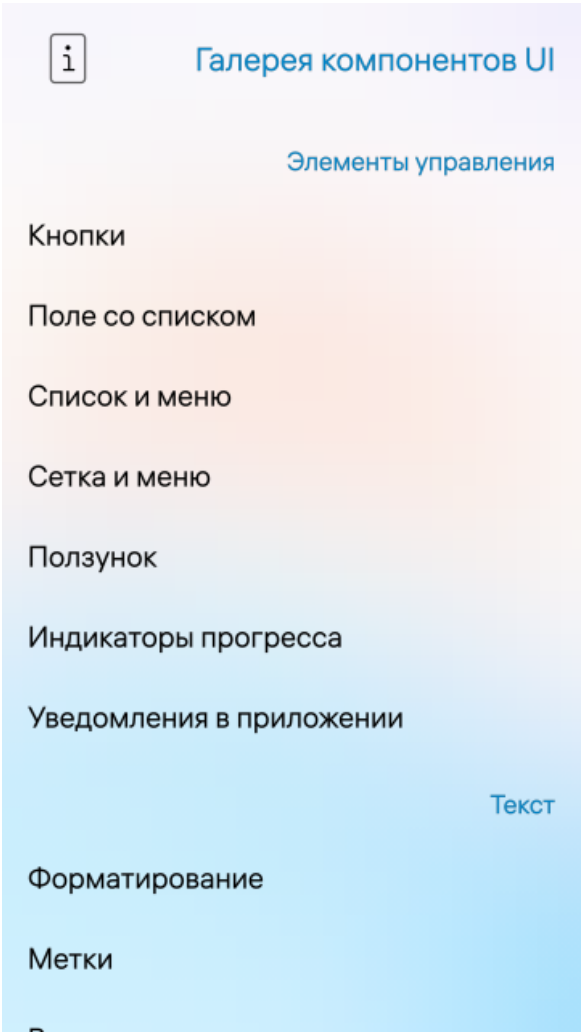

Рисунок 1 — Галерея компонентов UI.

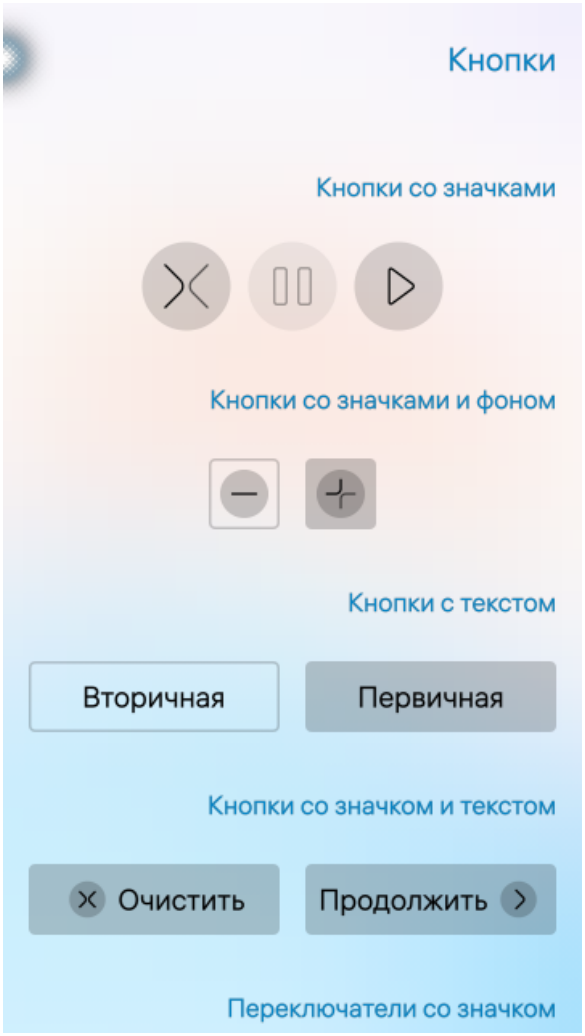

Рисунок 2 — Примеры кнопок.

```
Page {
  Button {
    anchors.centerIn: parent
    preferredWidth : Theme.buttonWidthMedium
    text: "Button"
    onClicked: console.log("Button clicked")
  }
}
```

Кнопка с полем значения:

- label : string — поле для краткого описания
- value : string — поле для значения
- description : string — поле для комментария

Пример такого приложения в соответствии с рисунком 2:

```
ValueButton {
  id: vb
  property int count: 0
  label: qsTr("Button")
  description: qsTr("Counting Value")
  value: count
  onClicked: vb.count++
}
```

Кнопка в виде иконки:

- icon : Image — изображение для кнопки
- highlighted : bool — подсвечена ли кнопка

Пример такого приложения в соответствии с рисунком 2:

```
IconButton {
  anchors.centerIn: parent
  icon.source: "image://theme/icon-m-play"
  icon.scale: 2
  onClicked: console.log("Button pressed")
}
```

Кнопка с переключателями:

- унаследован от компонента MouseArea
- automaticCheck : bool — автоматическое переключение при нажатии
- checked : bool — статус переключателя
- busy : bool — находится ли переключатель в состоянии «занят»
- icon : Image — иконка для переключателя
- TextSwitch — переключатель с полем для текста
  - text : string — текст рядом с индикатором статуса
  - description : string — описание переключателя
- IconTextSwitch — переключатель с иконкой и полем для текста

Пример такого приложения в соответствии с рисунком 2:

```
Switch {
  icon.source: "image://theme/icon-m-speaker-mute?" +
    (highlighted ? Theme.highlightColor :
    Theme.primaryColor)
}
```

```
TextSwitch {
  text: checked ? qsTr("Active") : qsTr("Inactive")
  description: qsTr("Switch with text label")
}
IconTextSwitch {
  icon.source: "image://theme/icon-m-speaker-mute?" +
    (highlighted ? Theme.highlightColor :
                   Theme.primaryColor)
  text: checked ? qsTr("Active") : qsTr("Inactive")
  description: qsTr("Switch with text label and icon")
}
```

MENUITEM и MENULABEL:

- MenuItem — пункт меню
  - text – текст пункта меню
  - color – цвет текст пункта меню
  - font – группа свойств для настройки шрифта
  - horizontalAlignment – горизонтальное выравнивание текста
- MenuLabel — описание вытягиваемого меню
  - text – текст метки в меню
  - color – цвет текста метки в меню
  - verticalOffset – вертикальный отступ

Пример такого приложения в соответствии с рисунком 3 и в соответствии с рисунком 4:

```
PullDownMenu {
  MenuItem {
    text: qsTr("Option 1")
    onClicked: console.log(qsTr("Option"))
  MenuItem {
    text: qsTr("Option 2")
    onClicked: console.log(qsTr("Option"))
  MenuLabel {
    text: qsTr("Informational label")
} }
PushUpMenu {
  backgroundColor: "red"
  highlightColor: backgroundColor
```

```
14    MenuItem {
15      text: qsTr("Option 1")
16      onClicked: console.log(qsTr("Option")) }
17    MenuItem {
18      text: qsTr("Option 2")
19      onClicked: console.log(qsTr("Option")) }
20    MenuLabel {
21      text: qsTr("Informational label")
22  } }
```

Рисунок 3 — Меню в виде списка.

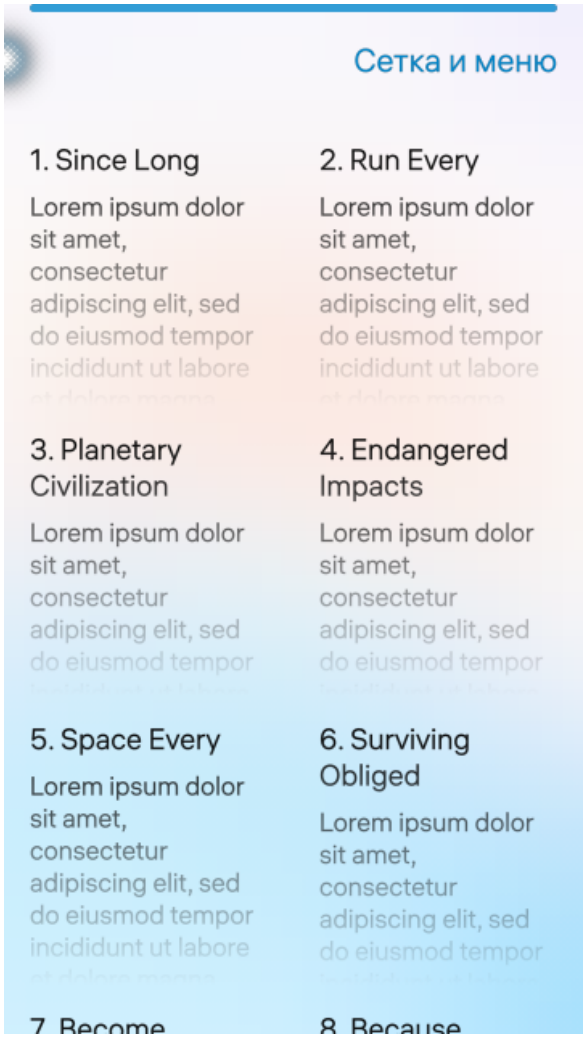

Рисунок 4 — Меню в виде сетки с комментариями.

В соответствии с рисунком 5 представлены виды ползунков и индикаторов, а в соответствии с рисунком 6 виды уведомлений в приложении.

В заключении рассмотрение и реализация различных компонентов UI Аврора в магистерской работе позволило сделать следующий вывод:

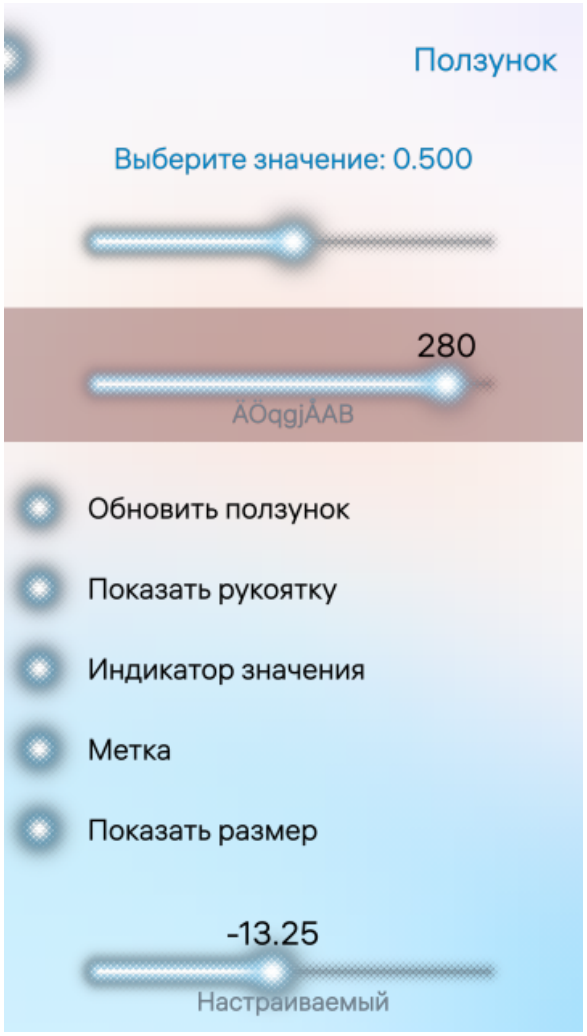

Рисунок 5 — Виды ползунков и индикаторов.

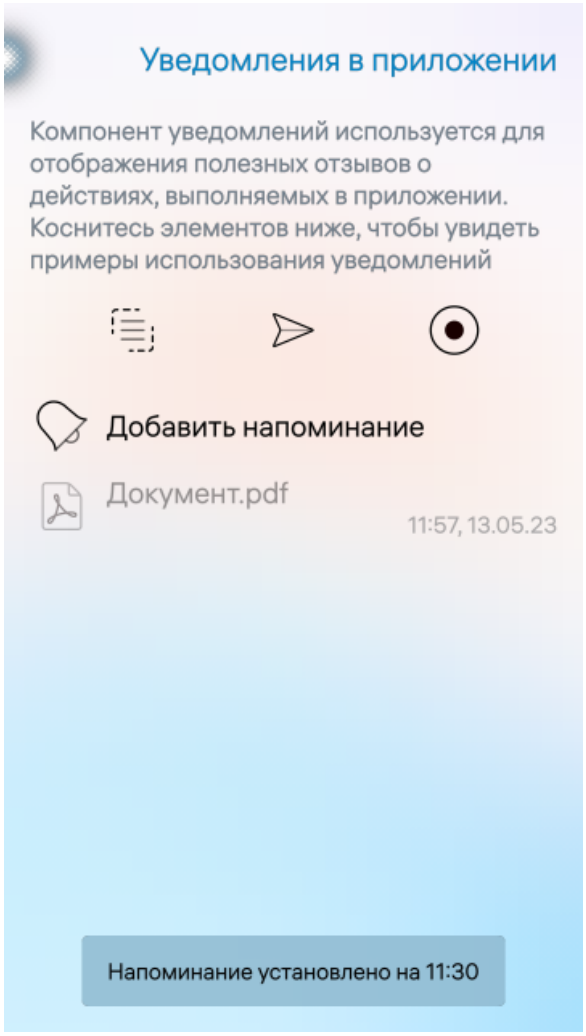

Рисунок 6 — Виды уведомлений в приложении.

- Простая для понимания структура приложений.
- Основная информация выделяется:
  - позицией;
  - цветом;
  - размером.
- Отзывчивый UI.
- Различные макеты для различных экранов.
- Автоматическая навигация для компонентов ввода.
- Для прокрутки всегда нужен индикатор данной области.