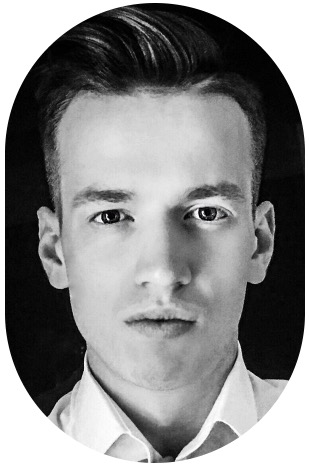
A11Y

МАСШТАБИРОВАНИЕ ШРИФТОВ В МОЙ МТС

Мой МТС



Тимофей Харитонов

iOS разработчик в Мой МТС

Улучшаю пользовательский опыт и работаю над персонализацией приложения. В коммерческой разработке с 2019 года

О МТС

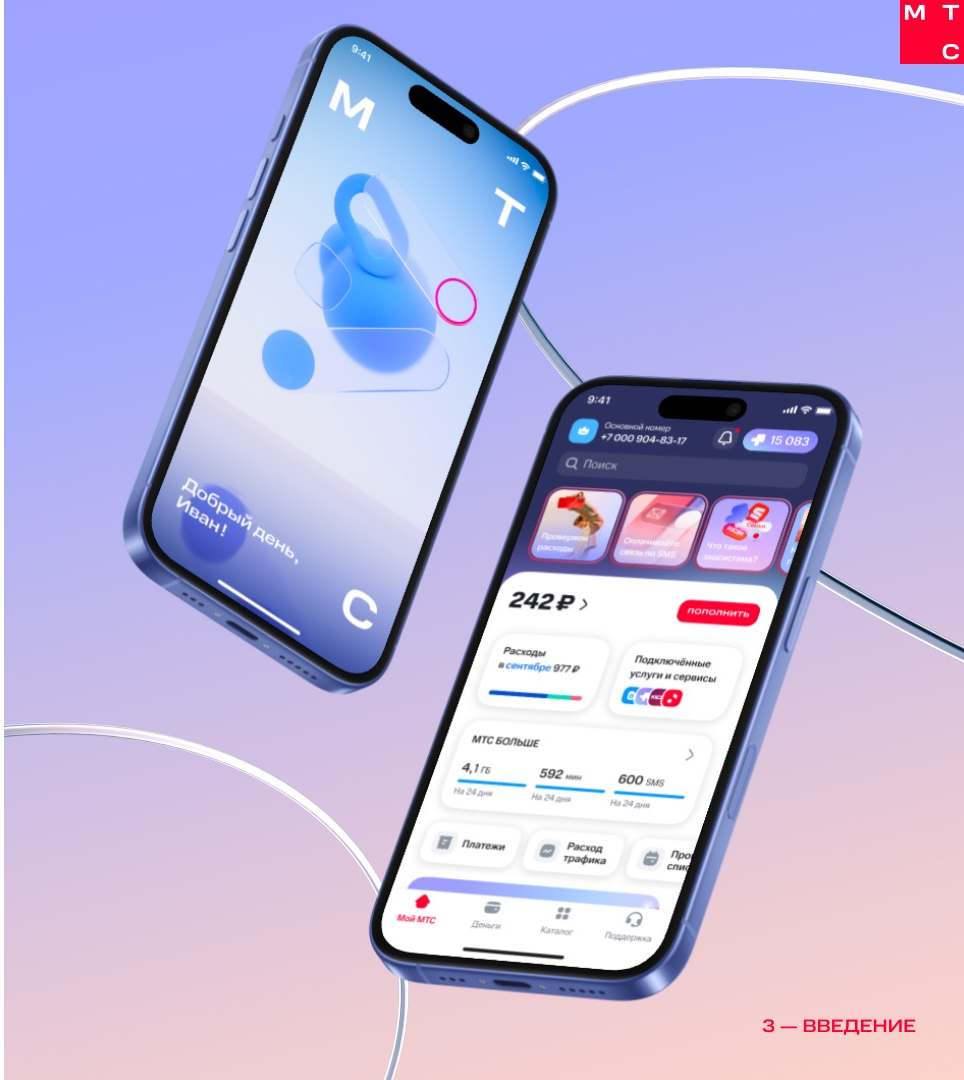
82.4 млн
абонентов
по всей стране
за 2024 г.

17.5 млн
экосистемные клиенты
пользуются не только услугами телекома
за 2024 г.

Мой МТС – это полноценный центр самообслуживания, поддержки клиента с возможностью бесшовно погружаться в экосистему

30 млн
пользователей
приложения «Мой МТС»
ежемесячных

4.7
средний рейтинг приложения в сторах
iOS и Android



ЦЕЛИ ДОКЛАДА

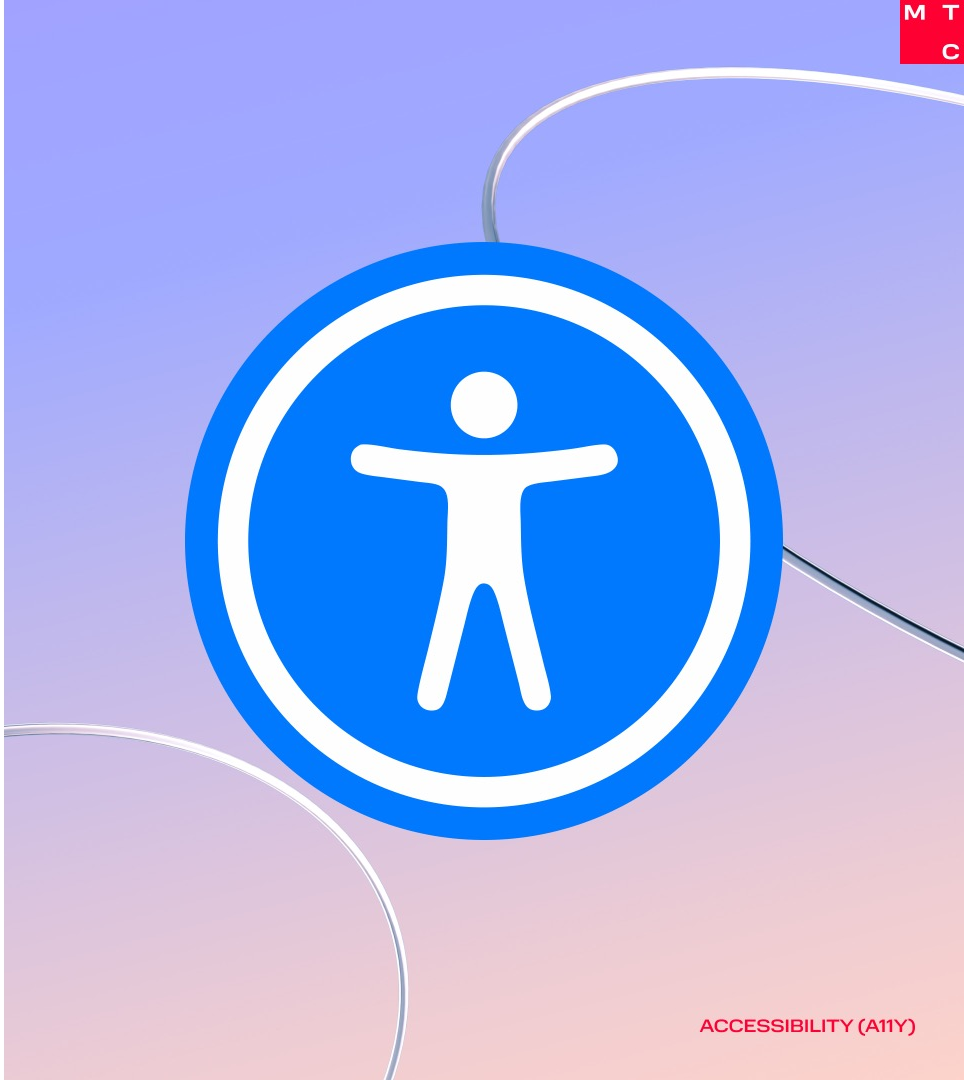
1. Accessibility, Dynamic Type
2. Жизненный цикл фичи
3. Аналитика и влияние на пользователей
4. Техническое решение
5. Открытые вопросы и итоги

Accessibility (A11Y)

ЦИФРОВАЯ ДОСТУПНОСТЬ

Доступные решения нужны каждому

Это концепция, которая помогает большинству пользователей, в том числе с особенностями здоровья, взаимодействовать с интерфейсом и технологиями



УРОВНИ ЦИФРОВОЙ ДОСТУПНОСТИ

A

Базовый

Интерфейсом неудобно пользоваться, а вспомогательные технологии работают только частично

AA

Идеальный

Интерфейс удобен для большинства пользователей

AAA

Специальный

Специализированная поддержка

ПРИНЦИПЫ ДОСТУПНОСТИ

Воспринимаемость

Пользователь воспринимает контент в интерфейсе любым доступным для него способом

Управляемость

Пользователь управляет контентом и интерфейсом с помощью любых доступных для него способов

Понятность

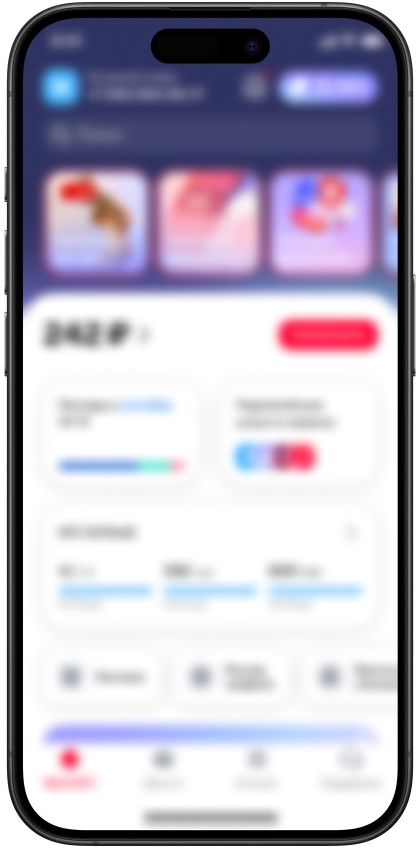
Пользователь понимает смысл контента и интерфейса вне зависимости от способа взаимодействия с ними

Устойчивость

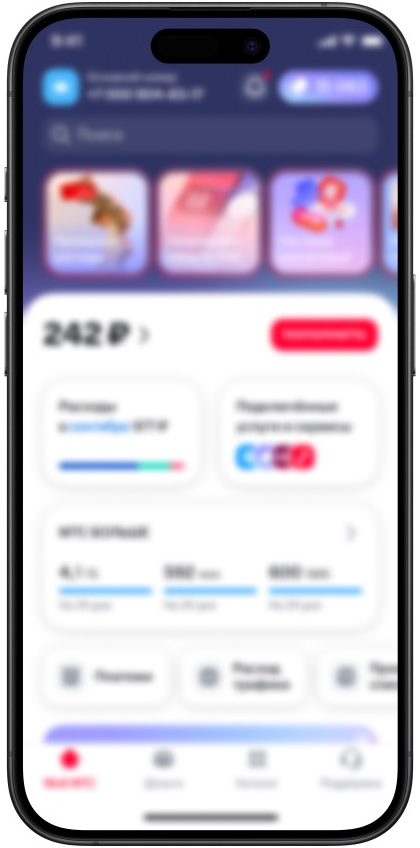
Интерфейс остается доступным при изменении своих версий, формата устройств или операционных систем

Масштабирование шрифтов

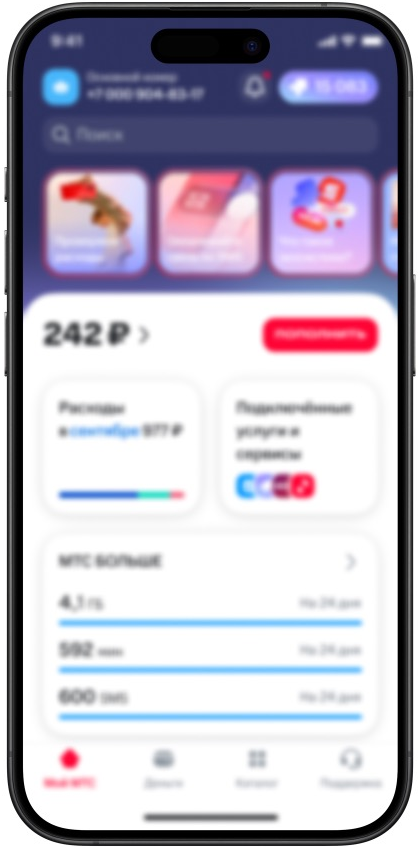
МАСШТАБИРОВАНИЕ ШРИФТОВ В МОЙ МТС



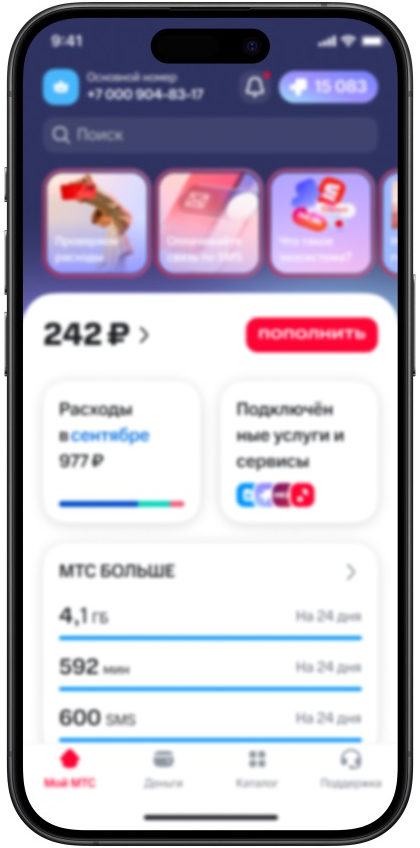
МАСШТАБИРОВАНИЕ ШРИФТОВ В МОЙ МТС



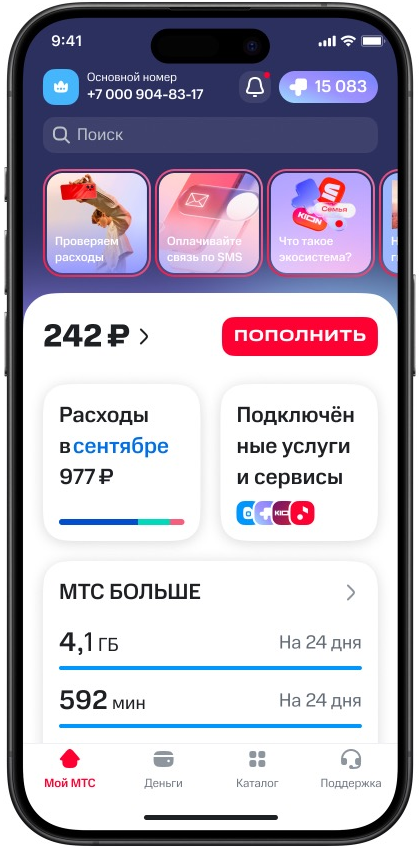
МАСШТАБИРОВАНИЕ ШРИФТОВ В МОЙ МТС



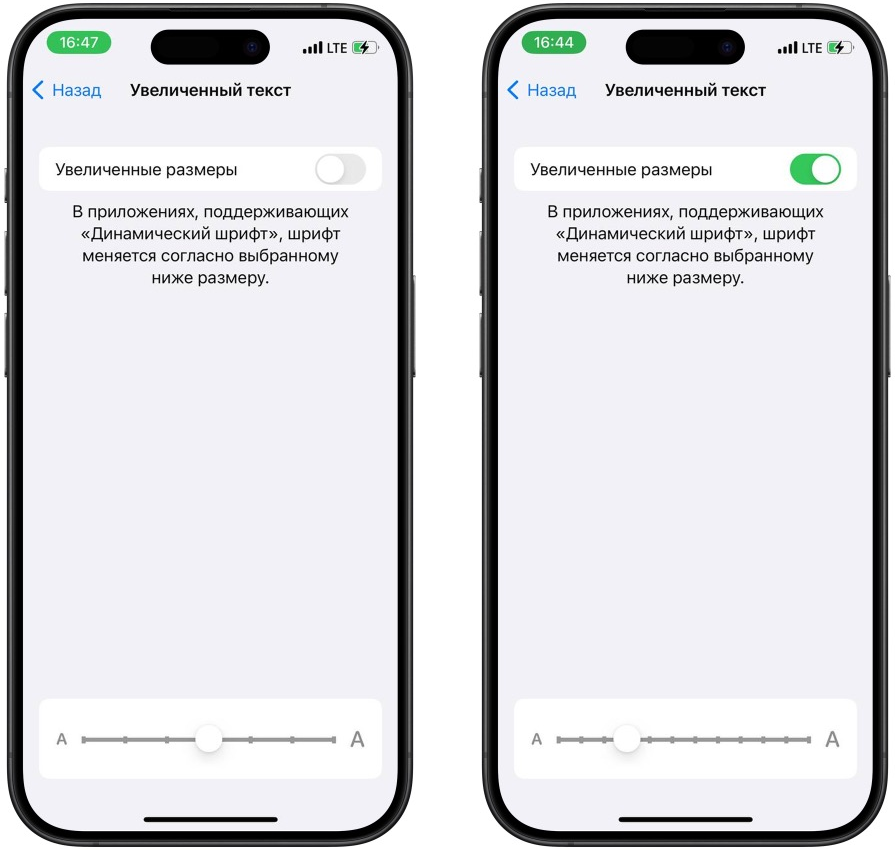
МАСШТАБИРОВАНИЕ ШРИФТОВ В МОЙ МТС



МАСШТАБИРОВАНИЕ ШРИФТОВ В МОЙ МТС



DYNAMIC TYPE: LARGER TEXT



ДЕМОНСТРАЦИЯ


Давай. Вошли и вышли, приключение
на 20 минут.


МТС
ЦИФРОВАЯ — ЭКОСИСТЕМА
МТС
ГАДЖЕТЫ
13:31
34
Анна Гаврилова
+7-915 115
1
ПОПОЛНИТЬ
Расходы
39,95 Р
Мои услути
SMART NONSTOP
9,46 гб
В 6 дней
224 мин
В 6 дней
400 SMS
В 6 дней
Мои МТС
Расходы
Каталог
Поддержка
СКОЛЬКО ?!??!??
Gx

Жизненный цикл фичи

КАК ПОЯВИЛАСЬ ИДЕЯ

«ЭТО НАДО ДЕЛАТЬ» — ДИЗАЙНЕРЫ

1. Темная тема
2. Масштабирование шрифтов
3. Скринридер (VoiceOver на iOS)
4. Другие аспекты

CUSTOMER EXPERIENCE (CX APP)

РЕСЕРЧ: ОСНОВНЫЕ ЦЕЛИ

1. Собрать аналитику
2. Выявить влияние на пользователей
3. Понять как масштабировать текст в приложении с более чем 500 экранов
4. Приоритизировать экраны и блоки

ИССЛЕДОВАНИЯ, НА КОТОРЫЕ ПОЛАГАЛИСЬ

ЯНДЕКС

inclusion.yandex.ru/
settingsresearch

APPT.ORG

appt.org/en/stats

ЕСЛИ БЫТЬ ТОЧНЫМ

techno.st/problems/
disability

ЦИФРЫ, НА КОТОРЫЕ ОРИЕНТИРОВАЛИСЬ

49 МЛН

людей в России

нуждаются в a11y-технологиях

51%

пользователей смартфонов

используют хотя бы одну настройку доступности

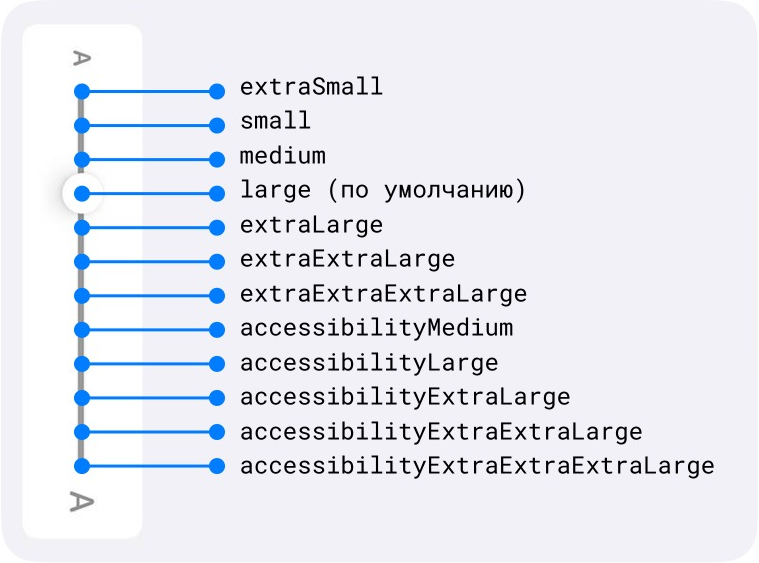
35%

пользователей

увеличивают или уменьшают размер шрифта

Сбор аналитики влияния на пользователей Мой МТС

КАТЕГОРИИ РАЗМЕРОВ ШРИФТОВ В ОС



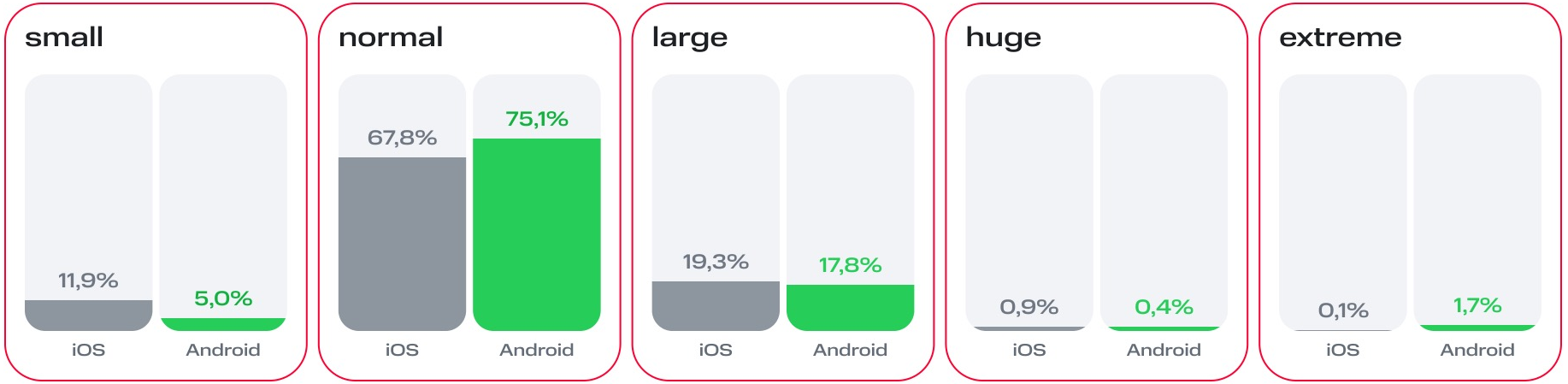
КАТЕГОРИИ РАЗМЕРОВ ШРИФТОВ В ОС

```
  small .extraSmall
  small .small
  small .medium
 normal .large (по умолчанию)
  large .extraLarge
  large .extraExtraLarge
  large .extraExtraExtraLarge
   huge .accessibilityMedium
   huge .accessibilityLarge
   huge .accessibilityExtraLarge
extreme .accessibilityExtraExtraLarge
extreme .accessibilityExtraExtraExtraLarge
```

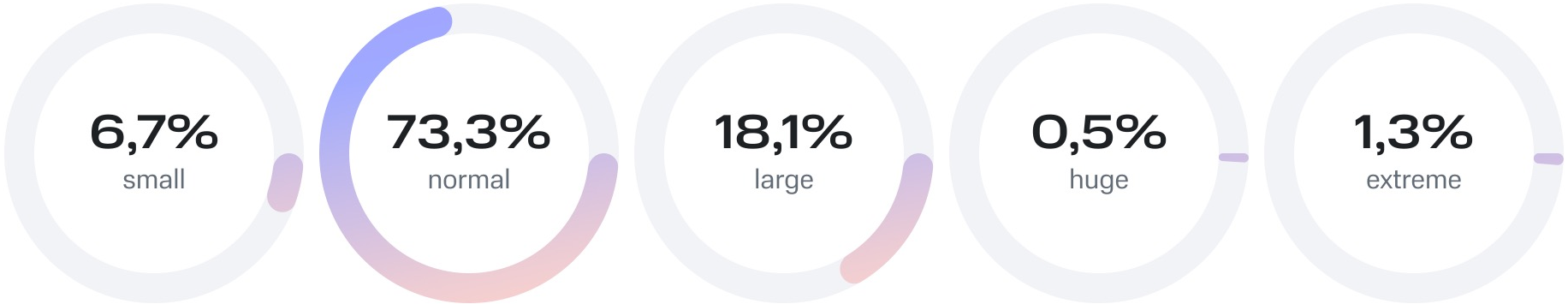
ТАБЛИЦА СООТВЕТСТВИЯ КАТЕГОРИЙ ШРИФТОВ

Категория шрифта	iOS	Android (коэффициенты, font scale factor)
small	extraSmall, small, medium	от 0.80
normal (default)	large	от 1.1 (1.5)
large	extraLarge, extraExtraLarge, extraExtraExtraLarge	от 1.2 до 1.5
huge	accessibilityMedium, accessibilityLarge, accessibilityExtraLarge	от 1.6 до 1.7
extreme	accessibilityExtraExtraLarge, accessibilityExtraExtraExtraLarge	от 1.8 и больше

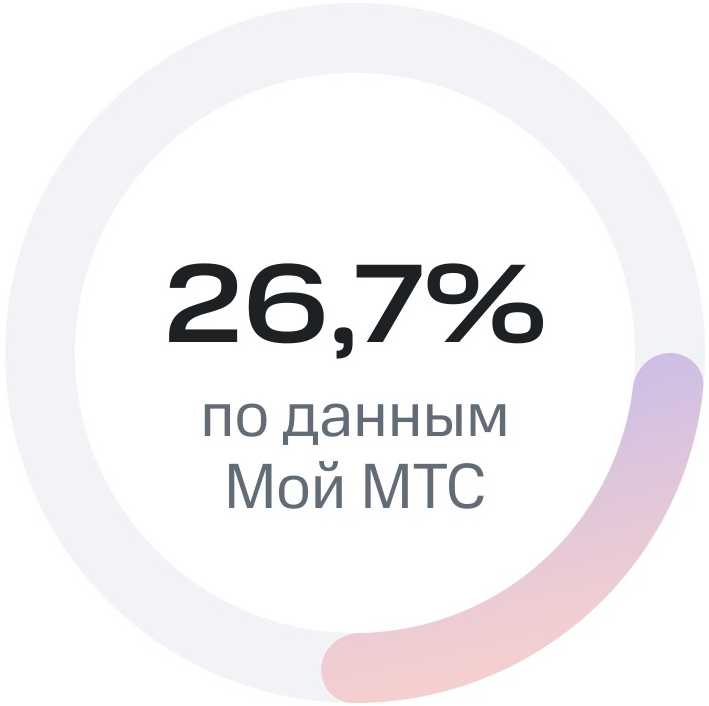
ВЛИЯНИЕ НА ПОЛЬЗОВАТЕЛЕЙ МОЙ МТС НА IOS И ANDROID



ОБЩЕЕ ВЛИЯНИЕ НА ПОЛЬЗОВАТЕЛЕЙ МОЙ МТС



ВЛИЯНИЕ НА ПОЛЬЗОВАТЕЛЕЙ



ПРЕЗЕНТАЦИЯ ФИЧИ БИЗНЕСУ

АРГУМЕНТЫ НА ЗАЩИТЕ

Цифры из исследований

Довольный клиент

Цифры влияния на наших пользователей

Современный тренд персонализации и кастомизации приложений

Новые технологии

Анализ конкурентов

Повышение уровня цифровой доступности

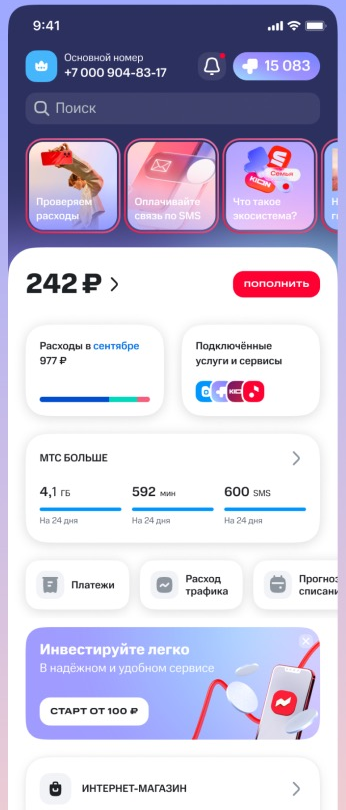
Directed by
ROBERT B. WEIDE

SDK DESIGN SYSTEM

МАКЕТЫ ГЛАВНОГО ЭКРАНА

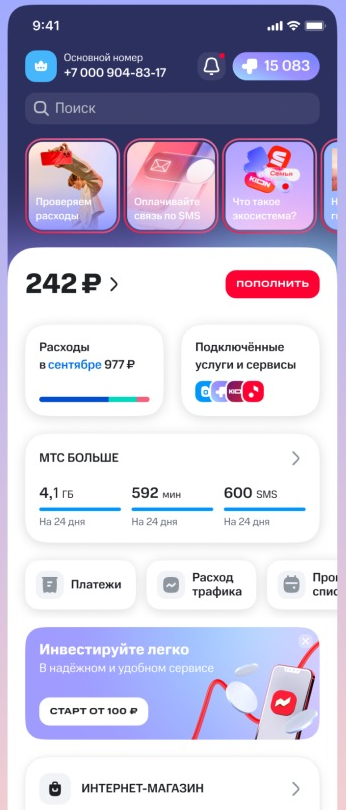
Small

Android Small, iOS Medium и ниже



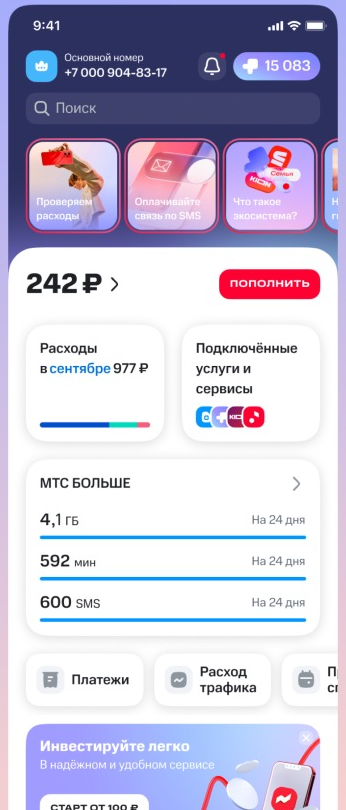
Normal

Android Normal, iOS Large



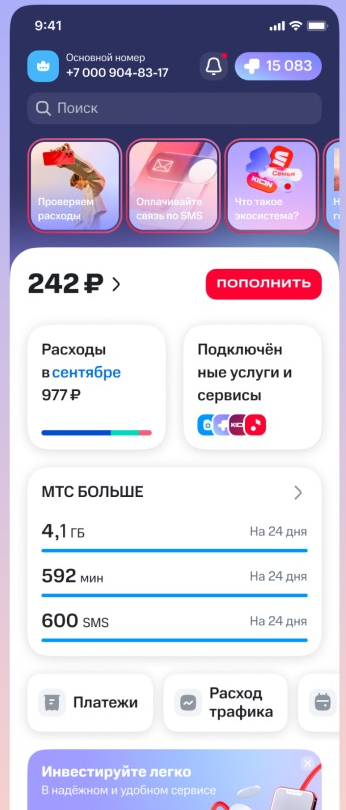
Large

Android Large, iOS xLarge



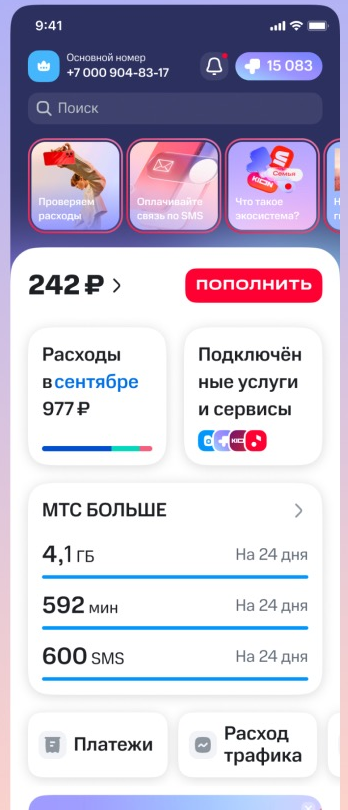
Huge

Android Huge, iOS xxLarge и выше



Extreme

Android Extreme, iOS AX5



ГЛОБАЛЬНЫЙ И ЛОКАЛЬНЫЙ ФЛАГ

1. Масштабирование всего приложения
2. Постепенное масштабирование

ДЕМОНСТРАЦИЯ

МАСШТАБИРОВАНИЕ ВСЕГО ПРИЛОЖЕНИЯ: ГЛОБАЛЬНЫЙ ФЛАГ

UIFont+Registration.swift

```swift
// Пример регистрации шрифтов при старте приложения с глобальным флагом масштабирования шрифтов:
// UIFont.registerFonts(isDynamic: true)

extension UIFont {

    /// Глобальный флаг масштабирования шрифта во всем приложение
    static var isDynamicFontEnabled: Bool = false

    /// Регистрация шрифтов при старте приложения
    static func registerFonts(isDynamic: Bool? = nil) {
        if let isDynamic {
            isDynamicFontEnabled = isDynamic

            // Даем разрешение текстовым объектам обновлять свой шрифт при изменении категории размера шрифта
            UILabel.appearance().adjustsFontForContentSizeCategory = true
            UITextView.appearance().adjustsFontForContentSizeCategory = true
            UITextField.appearance().adjustsFontForContentSizeCategory = true
            UIButton.appearance().adjustsImageSizeForAccessibilityContentSizeCategory = true
        }

        FontsLoader.registerFonts
    }
}
```

ФУНКЦИЯ РЕГИСТРАЦИИ ШРИФТОВ

UIFont+Registration.swift

```swift
// Пример регистрации шрифтов при старте приложения с глобальным флагом масштабирования шрифтов:
// UIFont.registerFonts(isDynamic: true)

extension UIFont {

    /// Глобальный флаг масштабирования шрифта во всем приложение
    static var isDynamicFontEnabled: Bool = false

    /// Регистрация шрифтов при старте приложения
    static func registerFonts(isDynamic: Bool? = nil) {
        if let isDynamic {
            isDynamicFontEnabled = isDynamic

            // Даем разрешение текстовым объектам обновлять свой шрифт при изменении категории размера шрифта
            UILabel.appearance().adjustsFontForContentSizeCategory = true
            UITextView.appearance().adjustsFontForContentSizeCategory = true
            UITextField.appearance().adjustsFontForContentSizeCategory = true
            UIButton.appearance().adjustsImageSizeForAccessibilityContentSizeCategory = true
        }

        FontsLoader.registerFonts
    }
}
```

РАЗРЕШАЕМ МАСШТАБИРОВАТЬСЯ ОБЪЕКТАМ

UIFont+Registration.swift

```swift
// Пример регистрации шрифтов при старте приложения с глобальным флагом масштабирования шрифтов:
// UIFont.registerFonts(isDynamic: true)

extension UIFont {

    /// Глобальный флаг масштабирования шрифта во всем приложение
    static var isDynamicFontEnabled: Bool = false

    /// Регистрация шрифтов при старте приложения
    static func registerFonts(isDynamic: Bool? = nil) {
        if let isDynamic {
            isDynamicFontEnabled = isDynamic

            // Даем разрешение текстовым объектам обновлять свой шрифт при изменении категории размера шрифта
            UILabel.appearance().adjustsFontForContentSizeCategory = true
            UITextView.appearance().adjustsFontForContentSizeCategory = true
            UITextField.appearance().adjustsFontForContentSizeCategory = true
            UIButton.appearance().adjustsImageSizeForAccessibilityContentSizeCategory = true
        }

        FontsLoader.registerFonts
    }
}
```

СОЗДАНИЕ ШРИФТА / ЛОКАЛЬНЫЙ ФЛАГ

UIFont+Extensions.swift

```swift
// Пример создания шрифта:
// label.font = UIFont.customFont(style: .headlineBold, isDynamic: true)

extension UIFont {

    /// *Название вашего кастомного шрифта*
    ///
    /// - Parameter style: Стиль типографики
    /// - Parameter isDynamic: Масштабирование шрифта (приоритетнее isDynamicFontEnabled)
    /// - Returns: UIFont
    static func customFont(style: CustomFontStyle, isDynamic: Bool? = nil) -> UIFont {
        let fontParams = style.parameters()
        let fontName = fontParams.fontName()
        let fontSize = fontParams.size
        let isDynamicFont = isDynamic ?? isDynamicFontEnabled

        guard let font = UIFont(name: fontName, size: fontSize) else {
            return UIFont.systemFont(ofSize: fontSize)
        }

        return isDynamicFont ? scaledFont(for: font, params: fontParams) : font
    }

    /// Масштабирование шрифта
    private static func scaledFont(for font: UIFont, params: CustomFontStyle.Parameters) -> UIFont {
        let metrics = UIFontMetrics(forTextStyle: params.appleTextStyle)
        let scaledFont = metrics.scaledFont(for: font, maximumPointSize: params.maxSize)

        return scaledFont
    }
}
```

СОЗДАНИЕ ШРИФТА / ЛОКАЛЬНЫЙ ФЛАГ

UIFont+Extensions.swift

```swift
// Пример создания шрифта:
// label.font = UIFont.customFont(style: .headlineBold, isDynamic: true)

extension UIFont {

    /// *Название вашего кастомного шрифта*
    ///
    /// - Parameter style: Стиль типографики
    /// - Parameter isDynamic: Масштабирование шрифта (приоритетнее isDynamicFontEnabled)
    /// - Returns: UIFont
    static func customFont(style: CustomFontStyle, isDynamic: Bool? = nil) -> UIFont {
        let fontParams = style.parameters()
        let fontName = fontParams.fontName()
        let fontSize = fontParams.size
        let isDynamicFont = isDynamic ?? isDynamicFontEnabled

        guard let font = UIFont(name: fontName, size: fontSize) else {
            return UIFont.systemFont(ofSize: fontSize)
        }

        return isDynamicFont ? scaledFont(for: font, params: fontParams) : font
    }

    /// Масштабирование шрифта
    private static func scaledFont(for font: UIFont, params: CustomFontStyle.Parameters) -> UIFont {
        let metrics = UIFontMetrics(forTextStyle: params.appleTextStyle)
        let scaledFont = metrics.scaledFont(for: font, maximumPointSize: params.maxSize)

        return scaledFont
    }
}
```

CustomFontStyle+Parameters.swift

```swift
extension CustomFontStyle {

    struct Parameters {

        enum Weight: String {
            case black = "Black"
            case bold = "Bold"
            case medium = "Medium"
            case regular = "Regular"
            case light = "Light"
        }

        // MARK: - Font parameters

        let size: CGFloat
        let maxSize: CGFloat // for a11y
        let lineHeight: CGFloat
        let maxLineHeight: CGFloat // for a11y
        let weight: Weight
        let appleTextStyle: UIFont.TextStyle // for a11y
        let appleTextStyleSUI: Font.TextStyle // for a11y

        private(set) var paragraphSpacing: CGFloat = 0
        private(set) var kernValue: CGFloat? = nil
        private(set) var baselineOffset: CGFloat = 0

        // MARK: - Functions

        func fontName() -> String {
            return "Roboto-" + weight.rawValue
        }
    }
}
```

СОЗДАНИЕ ШРИФТА / ЛОКАЛЬНЫЙ ФЛАГ

UIFont+Extensions.swift

```swift
// Пример создания шрифта:
// label.font = UIFont.customFont(style: .headlineBold, isDynamic: true)

extension UIFont {

    /// *Название вашего кастомного шрифта*
    ///
    /// - Parameter style: Стиль типографики
    /// - Parameter isDynamic: Масштабирование шрифта (приоритетнее isDynamicFontEnabled)
    /// - Returns: UIFont
    static func customFont(style: CustomFontStyle, isDynamic: Bool? = nil) -> UIFont {
        let fontParams = style.parameters()
        let fontName = fontParams.fontName()
        let fontSize = fontParams.size
        let isDynamicFont = isDynamic ?? isDynamicFontEnabled

        guard let font = UIFont(name: fontName, size: fontSize) else {
            return UIFont.systemFont(ofSize: fontSize)
        }

        return isDynamicFont ? scaledFont(for: font, params: fontParams) : font
    }

    /// Масштабирование шрифта
    private static func scaledFont(for font: UIFont, params: CustomFontStyle.Parameters) -> UIFont {
        let metrics = UIFontMetrics(forTextStyle: params.appleTextStyle)
        let scaledFont = metrics.scaledFont(for: font, maximumPointSize: params.maxSize)

        return scaledFont
    }
}
```

CustomFontStyle+Parameters.swift

```swift
extension CustomFontStyle {

    struct Parameters {

        enum Weight: String {
            case black = "Black"
            case bold = "Bold"
            case medium = "Medium"
            case regular = "Regular"
            case light = "Light"
        }

        // MARK: - Font parameters

        let size: CGFloat
        let maxSize: CGFloat // for a11y
        let lineHeight: CGFloat
        let maxLineHeight: CGFloat // for a11y
        let weight: Weight
        let appleTextStyle: UIFont.TextStyle // for a11y
        let appleTextStyleSUI: Font.TextStyle // for a11y

        private(set) var paragraphSpacing: CGFloat = 0
        private(set) var kernValue: CGFloat? = nil
        private(set) var baselineOffset: CGFloat = 0

        // MARK: - Functions

        func fontName() -> String {
            return "Roboto-" + weight.rawValue
        }
    }
}
```

СОЗДАНИЕ ШРИФТА / ЛОКАЛЬНЫЙ ФЛАГ

UIFont+Extensions.swift

```swift
// Пример создания шрифта:
// label.font = UIFont.customFont(style: .headlineBold, isDynamic: true)

extension UIFont {

    /// *Название вашего кастомного шрифта*
    ///
    /// - Parameter style: Стиль типографики
    /// - Parameter isDynamic: Масштабирование шрифта (приоритетнее isDynamicFontEnabled)
    /// - Returns: UIFont
    static func customFont(style: CustomFontStyle, isDynamic: Bool? = nil) -> UIFont {
        let fontParams = style.parameters()
        let fontName = fontParams.fontName()
        let fontSize = fontParams.size
        let isDynamicFont = isDynamic ?? isDynamicFontEnabled

        guard let font = UIFont(name: fontName, size: fontSize) else {
            return UIFont.systemFont(ofSize: fontSize)
        }

        return isDynamicFont ? scaledFont(for: font, params: fontParams) : font
    }

    /// Масштабирование шрифта
    private static func scaledFont(for font: UIFont, params: CustomFontStyle.Parameters) -> UIFont {
        let metrics = UIFontMetrics(forTextStyle: params.appleTextStyle)
        let scaledFont = metrics.scaledFont(for: font, maximumPointSize: params.maxSize)

        return scaledFont
    }
}
```

CustomFontStyle+Parameters.swift

```swift
extension CustomFontStyle {

    struct Parameters {

        enum Weight: String {
            case black = "Black"
            case bold = "Bold"
            case medium = "Medium"
            case regular = "Regular"
            case light = "Light"
        }

        // MARK: - Font parameters

        let size: CGFloat
        let maxSize: CGFloat // for a11y
        let lineHeight: CGFloat
        let maxLineHeight: CGFloat // for a11y
        let weight: Weight
        let appleTextStyle: UIFont.TextStyle // for a11y
        let appleTextStyleSUI: Font.TextStyle // for a11y

        private(set) var paragraphSpacing: CGFloat = 0
        private(set) var kernValue: CGFloat? = nil
        private(set) var baselineOffset: CGFloat = 0

        // MARK: - Functions

        func fontName() -> String {
            return "Roboto-" + weight.rawValue
        }
    }
}
```

МАСШТАБИРОВАНИЕ ШРИФТА

UIFont+Extensions.swift

```swift
// Пример создания шрифта:
// label.font = UIFont.customFont(style: .headlineBold, isDynamic: true)

extension UIFont {

    /// *Название вашего кастомного шрифта*
    ///
    /// - Parameter style: Стиль типографики
    /// - Parameter isDynamic: Масштабирование шрифта (приоритетнее isDynamicFontEnabled)
    /// - Returns: UIFont
    static func customFont(style: CustomFontStyle, isDynamic: Bool? = nil) -> UIFont {
        let fontParams = style.parameters()
        let fontName = fontParams.fontName()
        let fontSize = fontParams.size
        let isDynamicFont = isDynamic ?? isDynamicFontEnabled

        guard let font = UIFont(name: fontName, size: fontSize) else {
            return UIFont.systemFont(ofSize: fontSize)
        }

        return isDynamicFont ? scaledFont(for: font, params: fontParams) : font
    }

    /// Масштабирование шрифта
    private static func scaledFont(for font: UIFont, params: CustomFontStyle.Parameters) -> UIFont {
        let metrics = UIFontMetrics(forTextStyle: params.appleTextStyle)
        let scaledFont = metrics.scaledFont(for: font, maximumPointSize: params.maxSize)

        return scaledFont
    }
}
```

МАСШТАБИРОВАНИЕ ШРИФТА

UIFont+Extensions.swift

```swift
// Пример создания шрифта:
// label.font = UIFont.customFont(style: .headlineBold, isDynamic: true)

extension UIFont {

    /// *Название вашего кастомного шрифта*
    ///
    /// - Parameter style: Стиль типографики
    /// - Parameter isDynamic: Масштабирование шрифта (приоритетнее isDynamicFontEnabled)
    /// - Returns: UIFont
    static func customFont(style: CustomFontStyle, isDynamic: Bool? = nil) -> UIFont {
        let fontParams = style.parameters()
        let fontName = fontParams.fontName()
        let fontSize = fontParams.size
        let isDynamicFont = isDynamic ?? isDynamicFontEnabled

        guard let font = UIFont(name: fontName, size: fontSize) else {
            return UIFont.systemFont(ofSize: fontSize)
        }

        return isDynamicFont ? scaledFont(for: font, params: fontParams) : font
    }

    /// Масштабирование шрифта
    private static func scaledFont(for font: UIFont, params: CustomFontStyle.Parameters) -> UIFont {
        let metrics = UIFontMetrics(forTextStyle: params.appleTextStyle)
        let scaledFont = metrics.scaledFont(for: font, maximumPointSize: params.maxSize)

        return scaledFont
    }
}
```

МАСШТАБИРОВАНИЕ ШРИФТА

UIFont+Extensions.swift

```swift
// Пример создания шрифта:
// label.font = UIFont.customFont(style: .headlineBold, isDynamic: true)

extension UIFont {

    /// *Название вашего кастомного шрифта*
    ///
    /// - Parameter style: Стиль типографики
    /// - Parameter isDynamic: Масштабирование шрифта (приоритетнее isDynamicFontEnabled)
    /// - Returns: UIFont
    static func customFont(style: CustomFontStyle, isDynamic: Bool? = nil) -> UIFont {
        let fontParams = style.parameters()
        let fontName = fontParams.fontName()
        let fontSize = fontParams.size
        let isDynamicFont = isDynamic ?? isDynamicFontEnabled

        guard let font = UIFont(name: fontName, size: fontSize) else {
            return UIFont.systemFont(ofSize: fontSize)
        }

        return isDynamicFont ? scaledFont(for: font, params: fontParams) : font
    }

    /// Масштабирование шрифта
    private static func scaledFont(for font: UIFont, params: CustomFontStyle.Parameters) -> UIFont {
        let metrics = UIFontMetrics(forTextStyle: params.appleTextStyle)
        let scaledFont = metrics.scaledFont(for: font, maximumPointSize: params.maxSize)

        return scaledFont
    }
}
```

СОПОСТАВЛЕНИЕ СТИЛЯ МТС С СИСТЕМНЫМ

MTS DESIGN SYSTEM

iOS DYNAMIC TYPE SIZES

Promo	
Promo/Promo2 Short Wide	Large Title
Promo/Promo2 Short Wide	Large Title
Header	
Header/H1 36/40 Wide	Title 1
Header/H1 36/40 Comp	Title 1
Header/H2 32/36 Wide	Title 2
Header/H2 32/36 Comp	Title 2
Header/H3 24/28 Wide	Title 3
Header/H3 24/28 Comp	Title 3
Header/H3 24/28 Text	Title 3
Header/H4 20/24 Wide	Headline
Header/H4 20/24 Comp	Headline
Paragraph/P3 17/24 Medium Comp	Body
Paragraph/P3 17/24 Medium Text	Body
Paragraph/P3 17/24 Regular Comp	Body
Paragraph/P3 17/24 Regular Text	Body
Paragraph/P4 14/20 Bold Comp	Footnote
Paragraph/P4 14/20 Medium Comp	Footnote
Paragraph/P4 14/20 Regular Comp	Footnote
Paragraph/P4 14/20 Regular Text	Footnote
Paragraph/P4 14/20 Medium Upp Comp	Footnote
Paragraph/P4 14/20 Medium Upp text	Footnote
Caption	
Caption/C1 12/16 Bold UPP wide	Caption 1
Caption/C1 12/16 Bold Comp	Caption 1
Caption/C1 12/16 Medium Comp	Caption 1
Caption/C1 12/16 Regular Comp	Caption 1
Caption/C1 12/16 Medium upp comp	Caption 1
Caption/C2 10/12 bold upp wide	Caption 2

СОПОСТАВЛЕНИЕ КАСТОМНОГО СТИЛЯ С СИСТЕМНЫМ

CustomFontStyle.swift

```swift
enum CustomFontStyle {

    case titleBlack
    case titleBold
    case titleMedium
    case body
    case headline
    case body
    case footnote
    case caption

    var systemTextStyle: UIFont.TextStyle {
      switch self {
          case .titleBlack, .titleBold, .titleMedium:
              return .largeTitle
          case .headline:
              return .headline
          case .body:
              return .body
          case .footnote:
              return .footnote
          case .caption:
              return .caption1
        }
    }
}
```

ОГРАНИЧИВАЕМ РАЗМЕР ДЛЯ КАЖДОГО ШРИФТ-ТОКЕНА

UIFont+Extensions.swift

```swift
// Пример создания шрифта:
// label.font = UIFont.customFont(style: .headlineBold, isDynamic: true)

extension UIFont {

    /// *Название вашего кастомного шрифта*
    ///
    /// - Parameter style: Стиль типографики
    /// - Parameter isDynamic: Масштабирование шрифта (приоритетнее isDynamicFontEnabled)
    /// - Returns: UIFont
    static func customFont(style: CustomFontStyle, isDynamic: Bool? = nil) -> UIFont {
        let fontParams = style.parameters()
        let fontName = fontParams.fontName()
        let fontSize = fontParams.size
        let isDynamicFont = isDynamic ?? isDynamicFontEnabled

        guard let font = UIFont(name: fontName, size: fontSize) else {
            return UIFont.systemFont(ofSize: fontSize)
        }

        return isDynamicFont ? scaledFont(for: font, params: fontParams) : font
    }

    /// Масштабирование шрифта
    private static func scaledFont(for font: UIFont, params: CustomFontStyle.Parameters) -> UIFont {
        let metrics = UIFontMetrics(forTextStyle: params.appleTextStyle)
        let scaledFont = metrics.scaledFont(for: font, maximumPointSize: params.maxSize)

        return scaledFont
    }
}
```

maximumPointSize

MTS DESIGN SYSTEM

SIZE (PX)

Promo	xSmall	Small	Medium	Large (Default)	xLarge	xxLarge	xxxLarge	AX1	AX2	AX3	AX4	AX5
Promo2	41	42	43	44	46	48	50	54	58	62	66	70
Header												
H1 Все стили	33	34	35	36	38	40	42	46	51	56	61	66
H2 Все стили	29	30	31	32	34	36	38	44	49	54	60	66
H3 Все стили	21	22	23	24	26	28	30	35	41	47	53	60
H4 Все стили	17	18	19	20	22	24	26	31	36	43	50	56
Paragraph												
P1 Все стили	21	22	23	24	26	28	30	35	40	47	54	60
P2 Все стили	17	18	19	20	22	22	24	29	34	41	48	54
P3 Все стили	14	15	16	17	19	21	23	28	33	40	47	51
P4 Все стили	11	12	13	14	16	18	20	24	28	34	39	45
Caption												

NSAttributedString

NSAttributedString+Extensions.swift

```swift
// Пример:
// NSAttributedString.customAttributedString("Текст", style: .headlineBold, isDynamic: true)

extension NSAttributedString {

    /// Аттрибутная строка с пользовательским стилем
    static func customAttributedString(_ string: String,
                                       style: CustomFontStyle,
                                       textAlignment: NSTextAlignment = .left,
                                       lineBreakMode: NSLineBreakMode = .byTruncatingTail,
                                       extraAttributes: [NSAttributedString.Key: Any]? = nil,
                                       isDynamic: Bool? = nil) -> NSAttributedString {

        var attributes = style.fontAttributes(textAlignment: textAlignment,
                                              lineBreakMode: lineBreakMode,
                                              isDynamic: isDynamic)

        if let extraAttributes {
            attributes.merge(extraAttributes) { current, _ in current }
        }

        return NSAttributedString(string: string, attributes: attributes)
    }
}
```

NSAttributedString

NSAttributedString+Extensions.swift

```swift
// Пример:
// NSAttributedString.customAttributedString("Текст", style: .headlineBold, isDynamic: true)

extension NSAttributedString {

    /// Аттрибутная строка с пользовательским стилем
    static func customAttributedString(_ string: String,
                                       style: CustomFontStyle,
                                       textAlignment: NSTextAlignment = .left,
                                       lineBreakMode: NSLineBreakMode = .byTruncatingTail,
                                       extraAttributes: [NSAttributedString.Key: Any]? = nil,
                                       isDynamic: Bool? = nil) -> NSAttributedString {

        var attributes = style.fontAttributes(textAlignment: textAlignment,
                                              lineBreakMode: lineBreakMode,
                                              isDynamic: isDynamic)

        if let extraAttributes {
            attributes.merge(extraAttributes) { current, _ in current }
        }

        return NSAttributedString(string: string, attributes: attributes)
    }
}
```

fontAttributes

CustomFontStyle.swift

```swift
enum CustomFontStyle {

    func fontAttributes(textAlignment: NSTextAlignment = .left,
                        lineBreakMode: NSLineBreakMode = .byTruncatingTail,
                        isDynamic: Bool? = nil) -> [NSAttributedString.Key: Any] {
        let params = parameters() // Параметры токена шрифта
        let isDynamicFont = isDynamic ?? UIFont.isDynamicFontEnabled

        let paragraphStyle = NSMutableParagraphStyle()
        paragraphStyle.minimumLineHeight = params.lineHeight
        paragraphStyle.maximumLineHeight = isDynamicFont ? params.maxLineHeight : params.lineHeight
        paragraphStyle.paragraphSpacing = params.paragraphSpacing
        paragraphStyle.lineBreakMode = lineBreakMode
        paragraphStyle.alignment = textAlignment

        var attrs: [NSAttributedString.Key: Any] = [
            .baselineOffset: params.baselineOffset,
            .paragraphStyle: paragraphStyle,
            .font: UIFont.customFont(style: self, isDynamic: isDynamic)
        ]

        if let kernValue = params.kernValue {
            attrs[.kern] = kernValue
        }

        return attrs
    }
}
```

fontAttributes

CustomFontStyle.swift

```swift
enum CustomFontStyle {

    func fontAttributes(textAlignment: NSTextAlignment = .left,
                        lineBreakMode: NSLineBreakMode = .byTruncatingTail,
                        isDynamic: Bool? = nil) -> [NSAttributedString.Key: Any] {
        let params = parameters() // Параметры токена шрифта
        let isDynamicFont = isDynamic ?? UIFont.isDynamicFontEnabled

        let paragraphStyle = NSMutableParagraphStyle()
        paragraphStyle.minimumLineHeight = params.lineHeight
        paragraphStyle.maximumLineHeight = isDynamicFont ? params.maxLineHeight : params.lineHeight
        paragraphStyle.paragraphSpacing = params.paragraphSpacing
        paragraphStyle.lineBreakMode = lineBreakMode
        paragraphStyle.alignment = textAlignment

        var attrs: [NSAttributedString.Key: Any] = [
            .baselineOffset: params.baselineOffset,
            .paragraphStyle: paragraphStyle,
            .font: UIFont.customFont(style: self, isDynamic: isDynamic)
        ]

        if let kernValue = params.kernValue {
            attrs[.kern] = kernValue
        }

        return attrs
    }
}
```

maxLineHeight

MTS DESIGN SYSTEM

LINE HEIGHT (PX)

Promo	xSmall	Small	Medium	Large (Default)	xLarge	xxLarge	xxxLarge	AX1	AX2	AX3	AX4	AX5
Promo2	41	42	43	44	46	49	51	55	60	64	69	74
Header												
H1 Все стили	37	38	39	40	43	45	47	52	57	63	68	74
H2 Все стили	33	34	35	36	38	40	42	49	55	60	67	74
H3 Все стили	25	26	27	28	30	32	34	40	46	53	60	67
H4 Все стили	21	22	23	24	26	28	31	36	42	50	58	64
Paragraph												
P1 Все стили	29	30	31	32	34	36	39	44	50	58	66	72
P2 Все стили	25	26	27	28	30	32	35	40	46	54	62	68
P3 Все стили	21	22	23	24	26	28	31	36	42	50	58	64
P4 Все стили	17	18	19	20	22	24	26	31	35	42	48	54
Caption												

ДЕМОНСТРАЦИЯ

A11Y & SwiftUI. ПРИМЕР

ExampleView.swift

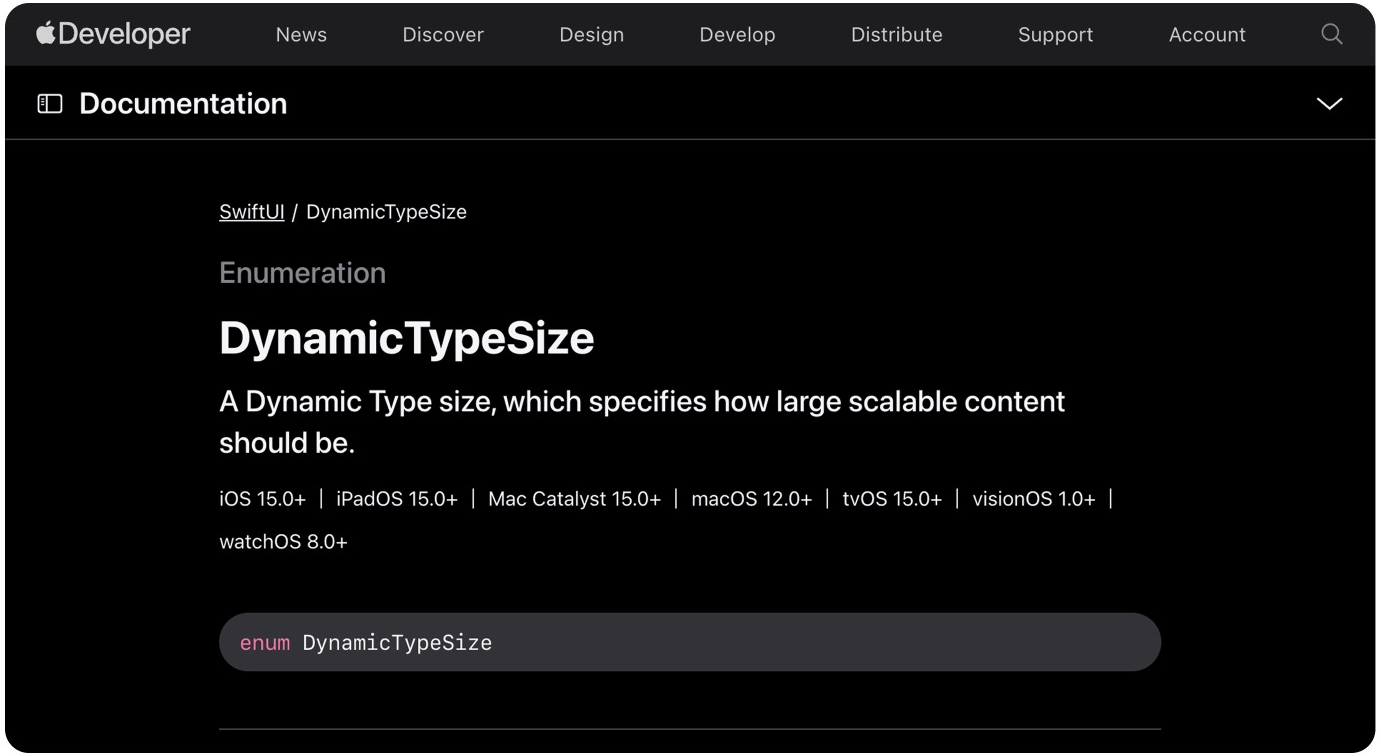
```swift
struct ExampleView: View {

    // Обновление вью при изменении категории размера шрифта
    @Environment(\.fontSizeCategory) private var fontSizeCategory

    var body: some View {
        Text("Размер категории шрифта: \(fontSizeCategory)")
            .font(.customFont(style: .bodyRegular, isDynamic: true))
    }
}
```

A11Y & SwiftUI. fontSizeCategory

EnvironmentValues+Extensions.swift

```swift
extension EnvironmentValues {

    var fontSizeCategory: AccessibilityFontSizeCategory {
        switch dynamicTypeSize {
        case .xSmall, .small, .medium:
            return .small
        case .large: // default apple font size category
            return .normal
        case .xLarge, .xxLarge, .xxxLarge:
            return .large
        case .accessibility1, .accessibility2, .accessibility3:
            return .huge
        case .accessibility4, .accessibility5:
            return .extreme
        default:
            return .unspecified
        }
    }
}
```

A11Y & SwiftUI. DynamicTypeSize



A11Y & SwiftUI. Font

Font+Extensions.swift

```swift
extension Font {

    static func customFont(style: CustomFontStyle, isDynamic: Bool? = nil) -> Font {
        return Font(UIFont.customFont(style: style, isDynamic: isDynamic))
    }
}
```

ДОРАБОТКА БЛОКОВ В МОЙ МТС

РЕГИСТРАЦИЯ ШРИФТОВ

AppDelegate.swift

```swift
class AppDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

        // Регистрируем шрифты из Дизайн-системы
        UIFont.registerFonts()

        // Даем разрешение текстовым объектам обновлять свой шрифт при изменении категории размера шрифта
        UILabel.appearance().adjustsFontForContentSizeCategory = true
        UITextField.appearance().adjustsFontForContentSizeCategory = true
        UITextView.appearance().adjustsFontForContentSizeCategory = true
        UIButton.appearance().adjustsImageSizeForAccessibilityContentSizeCategory = true
    }
}
```

РЕГИСТРАЦИЯ ШРИФТОВ

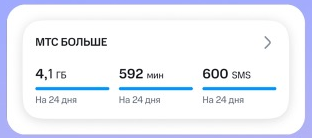
AppDelegate.swift

```swift
class AppDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

        // Регистрируем шрифты из Дизайн-системы
        UIFont.registerFonts()

        // Даем разрешение текстовым объектам обновлять свой шрифт при изменении категории размера шрифта
        UILabel.appearance().adjustsFontForContentSizeCategory = true
        UITextField.appearance().adjustsFontForContentSizeCategory = true
        UITextView.appearance().adjustsFontForContentSizeCategory = true
        UIButton.appearance().adjustsImageSizeForAccessibilityContentSizeCategory = true
    }
}
```

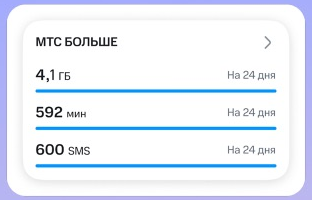
МАКЕТЫ БЛОКОВ

Small
Android Small, iOS Medium и ниже

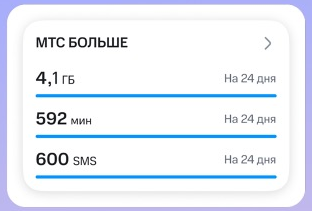
Normal
Android Normal, iOS Large



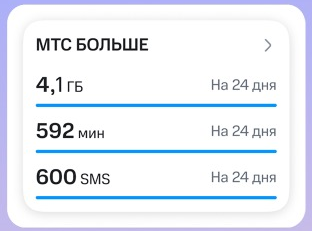
Large
Android Large, iOS xLarge



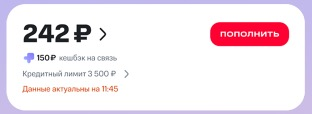
Huge
Android Huge, iOS xxLarge и выше



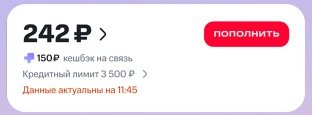
Extreme
Android Extreme, iOS AX5



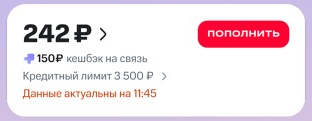
Small
Android Small, iOS Medium и ниже



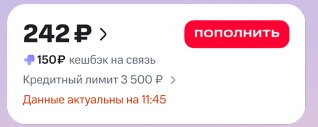
Normal
Android Normal, iOS Large



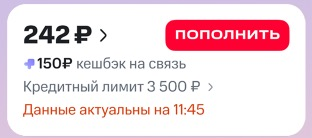
Large
Android Large, iOS xLarge



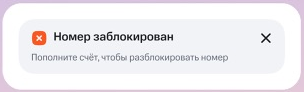
Huge
Android Huge, iOS xxLarge и выше



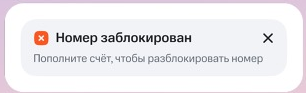
Extreme
Android Extreme, iOS AX5



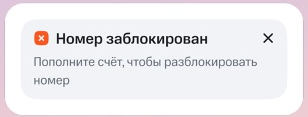
Small
Android Small, iOS Medium и ниже



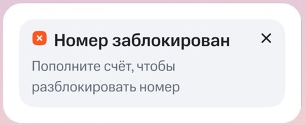
Normal
Android Normal, iOS Large



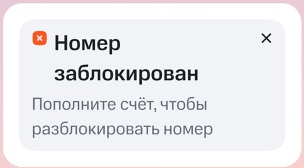
Large
Android Large, iOS xLarge



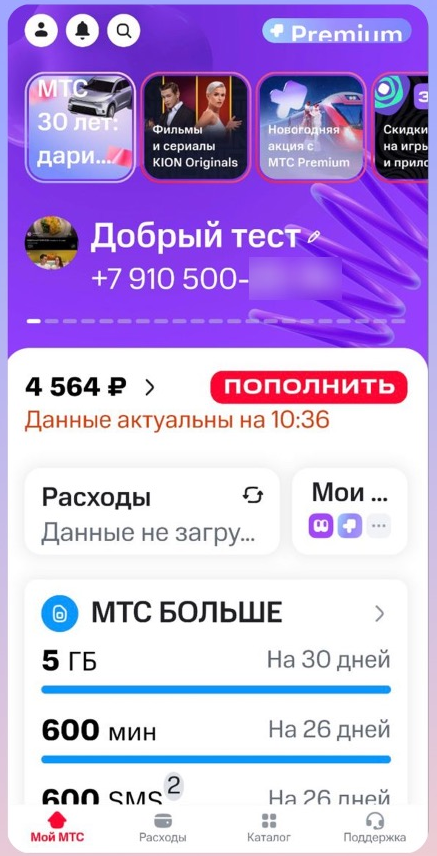
Huge
Android Huge, iOS xxLarge и выше

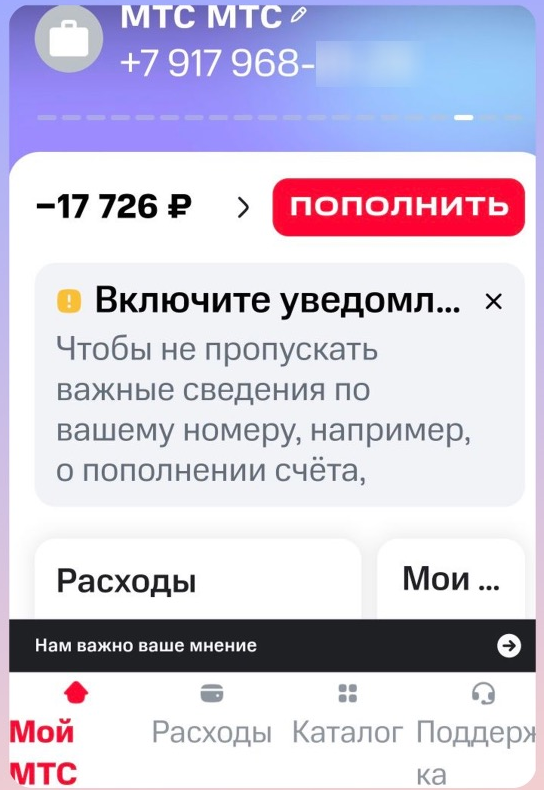


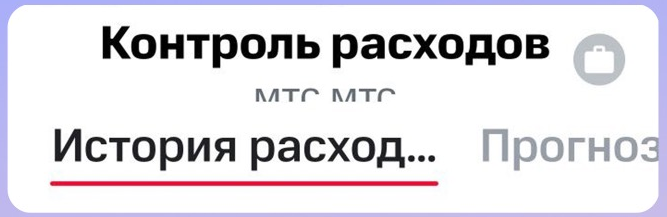
Extreme
Android Extreme, iOS AX5



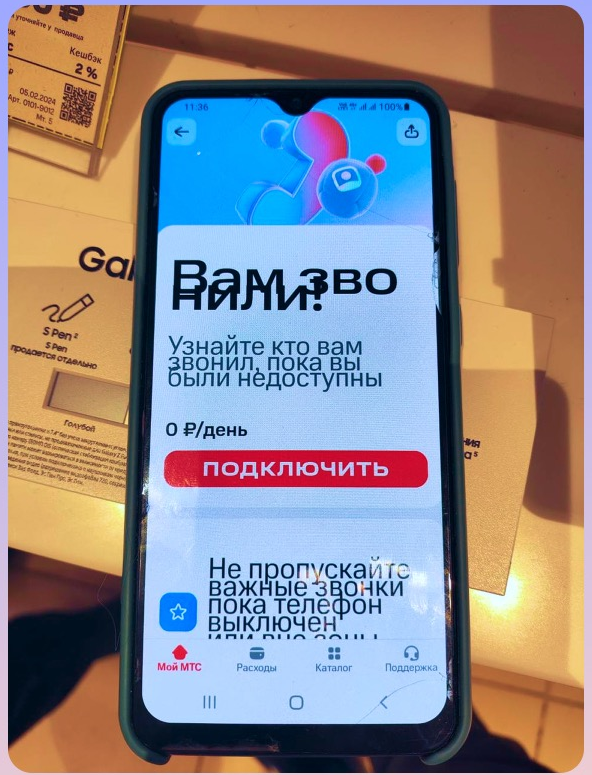
ПРОБЛЕМЫ С UI

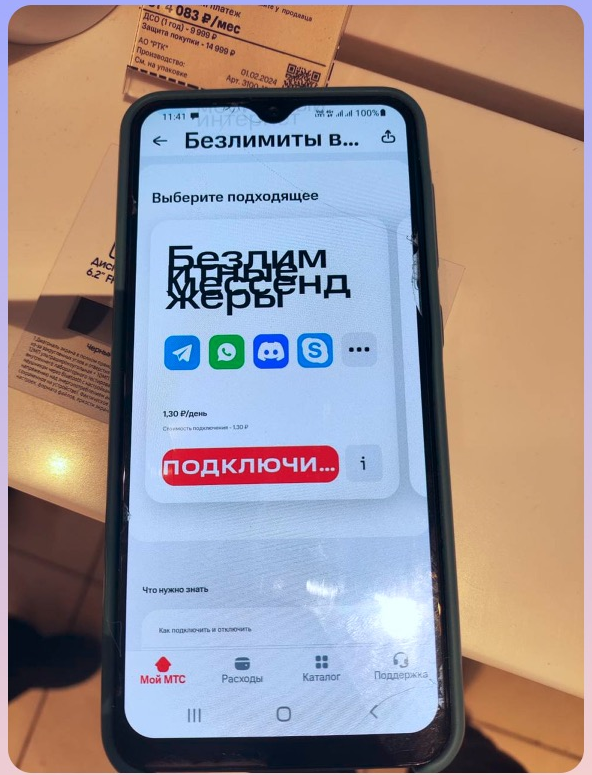




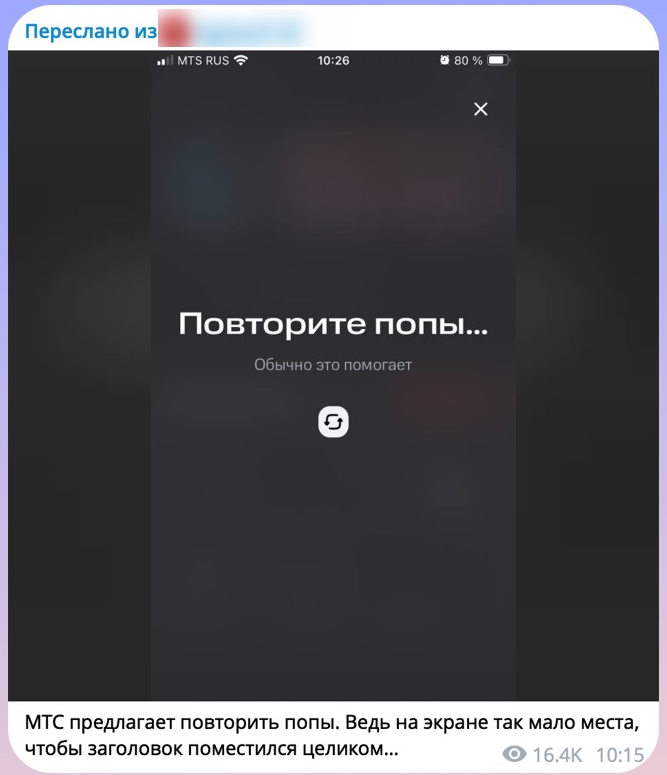


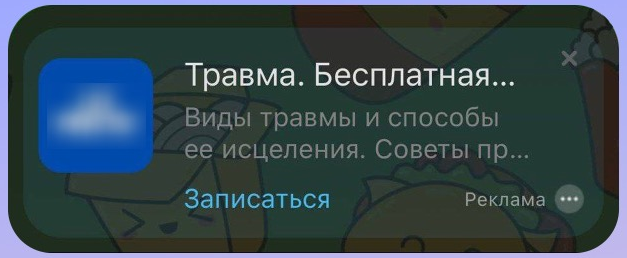
ПРОБЛЕМЫ С UI

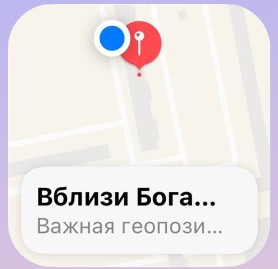


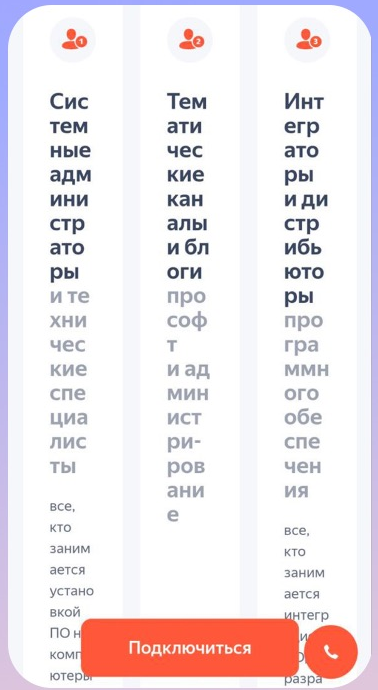


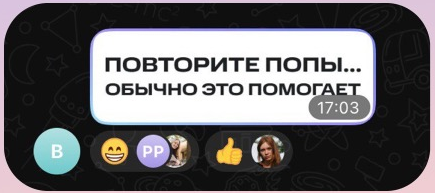
ПОЧЕМУ ВАЖНО ОБРАЩАТЬ ВНИМАНИЕ НА МНОГОТОЧИЯ

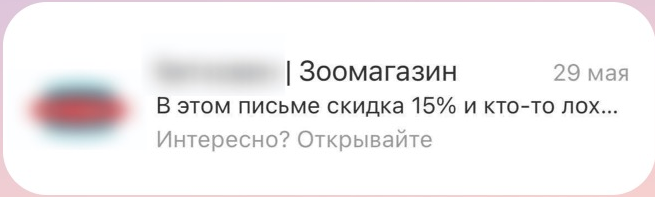












ПРИМЕР UIView. A11Y & numberOfLines

TestView.swift

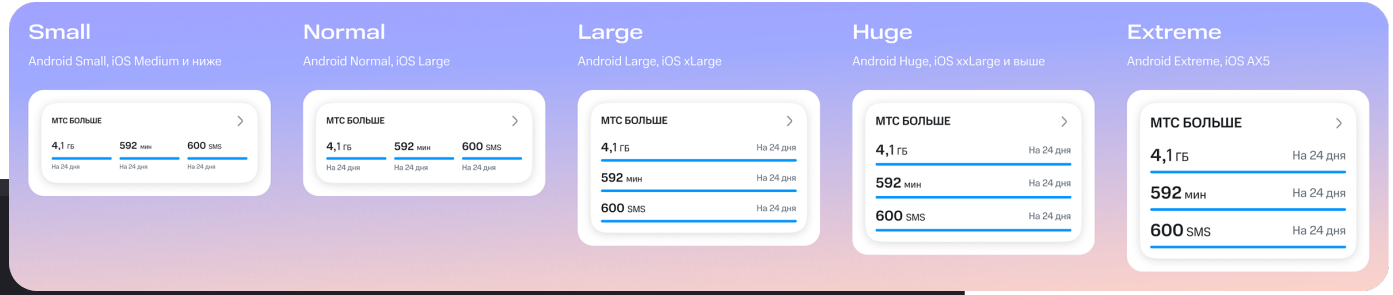
```swift
class TestView: UIView {

  private let titleLabel: UILabel = {
      let label = UILabel(font: customFont(style: .headlineBold, isDynamic: true), textColor: .green)
      label.text = "Mobius 2025 Spring"
      label.numberOfLines = 0
      return label
  }()
}
```

АДАПТАЦИЯ ТЕКСТА И КОНТЕНТА

ExampleView.swift

```swift
// UIKit
// buttonStackView.axis = fontSizeCategory > .normal ? .vertical : .horizontal

struct ExampleView: View {

    // Обновление вью при изменении категории размера шрифта
    @Environment(\.fontSizeCategory) private var fontSizeCategory

    var body: some View {
        Text("Размер категории шрифта: \(fontSizeCategory)")
            .font(.customFont(style: .bodyRegular, isDynamic: true))
            .foregroundStyle( fontSizeCategory > .normal ? .red : .white)
    }
}
```

АДАПТАЦИЯ ТЕКСТА И КОНТЕНТА



ExampleView.swift

```swift
// UIKit
// buttonStackView.axis = fontSizeCategory > .normal ? .vertical : .horizontal

struct ExampleView: View {

    // Обновление вью при изменении категории размера шрифта
    @Environment(\.fontSizeCategory) private var fontSizeCategory

    var body: some View {
        Text("Размер категории шрифта: \(fontSizeCategory)")
            .font(.customFont(style: .bodyRegular, isDynamic: true))
            .foregroundStyle( fontSizeCategory > .normal ? .red : .white)
    }
}
```

КАСТОМНЫЕ КАТЕГОРИИ РАЗМЕРА ШРИФТА

UITraitEnvironment+Extensions.swift

```swift
extension UITraitEnvironment {

    var fontSizeCategory: AccessibilityFontSizeCategory {
        switch traitCollection.preferredContentSizeCategory {
        case .extraSmall, .small, .medium:
            return .small
        case .large: // default apple font size category
            return .normal
        case .extraLarge, .extraExtraLarge, .extraExtraExtraLarge:
            return .large
        case .accessibilityMedium, .accessibilityLarge, .accessibilityExtraLarge:
            return .huge
        case .accessibilityExtraExtraLarge, .accessibilityExtraExtraExtraLarge:
            return .extreme
        default:
            return .unspecified
        }
    }
}
```

EnvironmentValues+Extensions.swift

```swift
extension EnvironmentValues {

    var fontSizeCategory: AccessibilityFontSizeCategory {
        switch dynamicTypeSize {
        case .xSmall, .small, .medium:
            return .small
        case .large: // default apple font size category
            return .normal
        case .xLarge, .xxLarge, .xxxLarge:
            return .large
        case .accessibility1, .accessibility2, .accessibility3:
            return .huge
        case .accessibility4, .accessibility5:
            return .extreme
        default:
            return .unspecified
        }
    }
```

AccessibilityFontSizeCategory.swift

```swift
enum AccessibilityFontSizeCategory: Comparable {
    case small
    case normal
    case large
    case huge
    case extreme
    case unspecified
}
```

КАСТОМНЫЕ КАТЕГОРИИ РАЗМЕРА ШРИФТА

UITraitEnvironment+Extensions.swift

```swift
extension UITraitEnvironment {

    var fontSizeCategory: AccessibilityFontSizeCategory {
        switch traitCollection.preferredContentSizeCategory {
        case .extraSmall, .small, .medium:
            return .small
        case .large: // default apple font size category
            return .normal
        case .extraLarge, .extraExtraLarge, .extraExtraExtraLarge:
            return .large
        case .accessibilityMedium, .accessibilityLarge, .accessibilityExtraLarge:
            return .huge
        case .accessibilityExtraExtraLarge, .accessibilityExtraExtraExtraLarge:
            return .extreme
        default:
            return .unspecified
        }
    }
}
```

EnvironmentValues+Extensions.swift

```swift
extension EnvironmentValues {

    var fontSizeCategory: AccessibilityFontSizeCategory {
        switch dynamicTypeSize {
        case .xSmall, .small, .medium:
            return .small
        case .large: // default apple font size category
            return .normal
        case .xLarge, .xxLarge, .xxxLarge:
            return .large
        case .accessibility1, .accessibility2, .accessibility3:
            return .huge
        case .accessibility4, .accessibility5:
            return .extreme
        default:
            return .unspecified
        }
    }
```

AccessibilityFontSizeCategory.swift

```swift
enum AccessibilityFontSizeCategory: Comparable {
    case small
    case normal
    case large
    case huge
    case extreme
    case unspecified
}
```

UIKit. traitCollectionDidChange (deprecated)

TestView.swift

```swift
class TestView: UIView {

    override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
        super.traitCollectionDidChange(previousTraitCollection)
        if #unavailable(iOS 18.0) {
            resizeContentSizeCategory(previousTraitCollection: previousTraitCollection)
        }
    }

    private func resizeContentSizeCategory(previousTraitCollection: UITraitCollection?) {
        if previousTraitCollection?.preferredContentSizeCategory != traitCollection.preferredContentSizeCategory {
            buttonStackView.axis = fontSizeCategory > .normal ? .vertical : .horizontal
        }
    }
}
```

traitCollectionDidChange(_:) Deprecated

Reports changes in the iOS interface environment.

iOS 8.0–17.0 Deprecated | iPadOS 8.0–17.0 Deprecated | Mac Catalyst 13.1–17.0 Deprecated | tvOS Deprecated | visionOS 1.0–1.0 Deprecated

```swift
@MainActor
func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?)
```

UIKit. registerForTraitChanges (iOS 18)

TestView.swift

```swift
class TestView: UIView {

    override func viewDidLoad() {
        super.viewDidLoad()

        registerTraitDidChange()
    }

    override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
        super.traitCollectionDidChange(previousTraitCollection)
        if #unavailable(iOS 18.0) {
            resizeContentSizeCategory(previousTraitCollection: previousTraitCollection)
        }
    }

    private func registerTraitDidChange() {
        if #available(iOS 18.0, *) {
            registerForTraitChanges([UITraitPreferredContentSizeCategory.self]) { (self: Self, previousTraitCollection: UITraitCollection) in
                self.resizeContentSizeCategory(previousTraitCollection: previousTraitCollection)
            }
        }
    }

    private func resizeContentSizeCategory(previousTraitCollection: UITraitCollection?) {
        if previousTraitCollection?.preferredContentSizeCategory != traitCollection.preferredContentSizeCategory {
            buttonStackView.axis = fontSizeCategory > .normal ? .vertical : .horizontal
        }
    }
}
```

UIKit. registerForTraitChanges (iOS 18)

TestView.swift

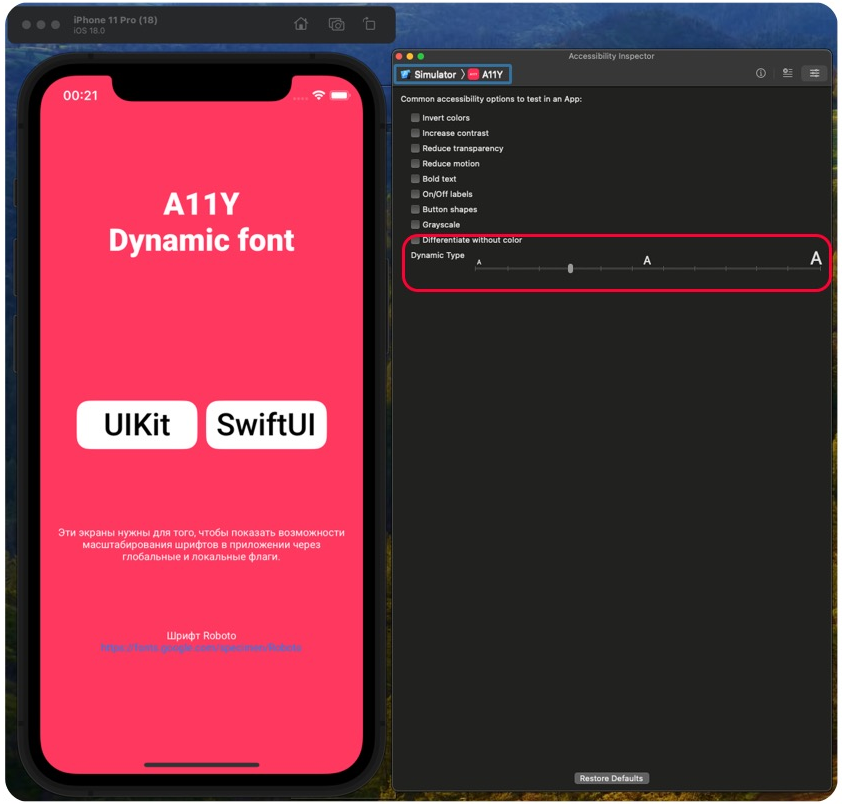
```swift
class TestView: UIView {

    override func viewDidLoad() {
        super.viewDidLoad()

        registerTraitDidChange()
    }

    override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
        super.traitCollectionDidChange(previousTraitCollection)
        if #unavailable(iOS 18.0) {
            resizeContentSizeCategory(previousTraitCollection: previousTraitCollection)
        }
    }

    private func registerTraitDidChange() {
        if #available(iOS 18.0, *) {
            registerForTraitChanges([UITraitPreferredContentSizeCategory.self]) { (self: Self, previousTraitCollection: UITraitCollection) in
                self.resizeContentSizeCategory(previousTraitCollection: previousTraitCollection)
            }
        }
    }

    private func resizeContentSizeCategory(previousTraitCollection: UITraitCollection?) {
        if previousTraitCollection?.preferredContentSizeCategory != traitCollection.preferredContentSizeCategory {
            buttonStackView.axis = fontSizeCategory > .normal ? .vertical : .horizontal
        }
    }
}
```

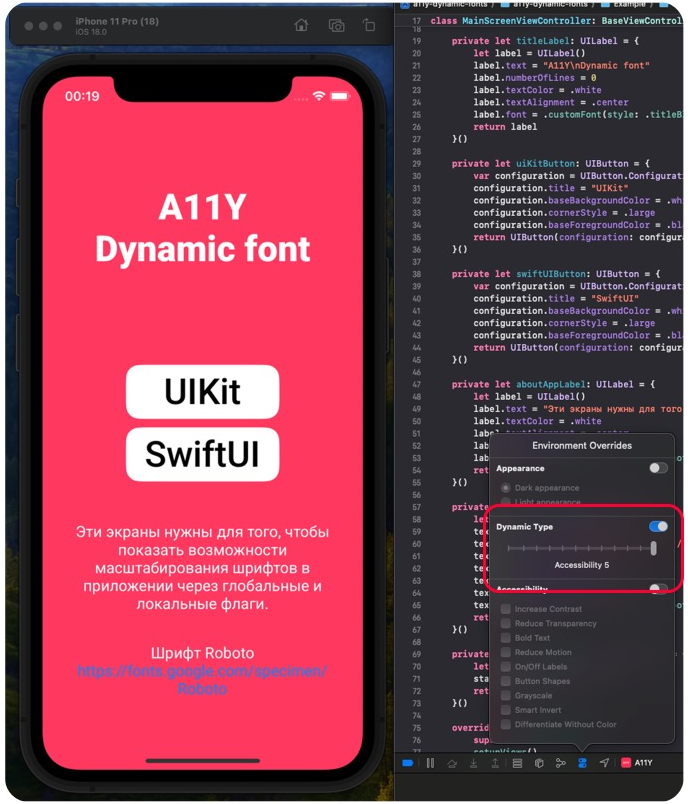
UIKit. registerForTraitChanges (iOS 18)

TestView.swift

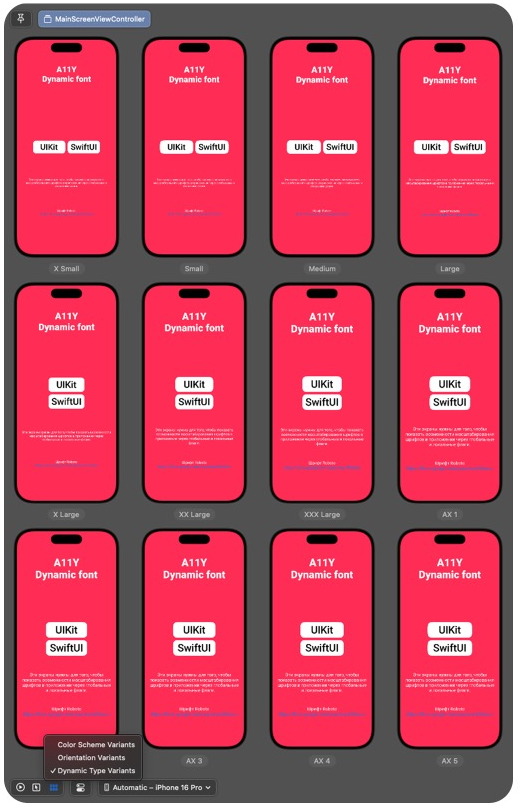
```swift
class TestView: UIView {

    override func viewDidLoad() {
        super.viewDidLoad()

        registerTraitDidChange()
    }

    override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
        super.traitCollectionDidChange(previousTraitCollection)
        if #unavailable(iOS 18.0) {
            resizeContentSizeCategory(previousTraitCollection: previousTraitCollection)
        }
    }

    private func registerTraitDidChange() {
        if #available(iOS 18.0, *) {
            registerForTraitChanges([UITraitPreferredContentSizeCategory.self]) { (self: Self, previousTraitCollection: UITraitCollection) in
                self.resizeContentSizeCategory(previousTraitCollection: previousTraitCollection)
            }
        }
    }

    private func resizeContentSizeCategory(previousTraitCollection: UITraitCollection?) {
        if previousTraitCollection?.preferredContentSizeCategory != traitCollection.preferredContentSizeCategory {
            buttonStackView.axis = fontSizeCategory > .normal ? .vertical : .horizontal
        }
    }
}
```

ТЕСТИРОВАНИЕ. A11Y Inspector & Environment Overrides





SwiftUI Preview



ПРОЦЕСС ТЕСТИРОВАНИЯ БЛОКОВ

1. Разработчик
2. Тестировщик
3. Дизайнер
4. Пользователь

РЕВЬЮ ОТ ДИЗАЙНА И ТЕСТИРОВАНИЕ

ЗАВЕРШЕНИЕ ЦИКЛА

Открытые вопросы

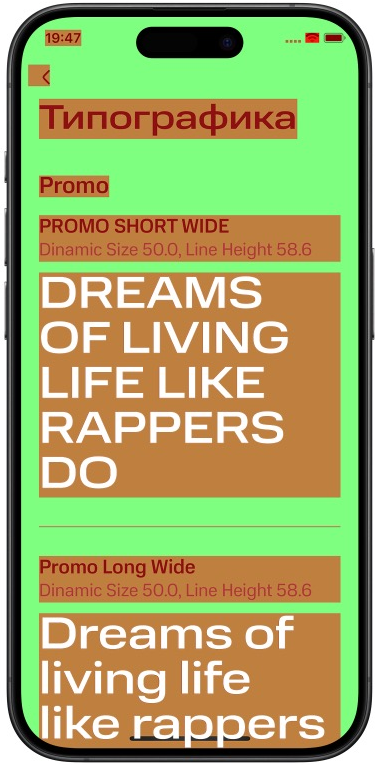
ОТКРЫТЫЕ ВОПРОСЫ

1. Доработка компонентов Дизайн-системы
2. Статичная верстка
3. Текст на изображениях
4. WebView / кроссплатформа и прочие технологии
5. Межстрочный интервал в NSAttributedString

РАЗРЕЖЕННОСТЬ МЕЖСТРОЧНОГО ИНТЕРВАЛА

small

normal



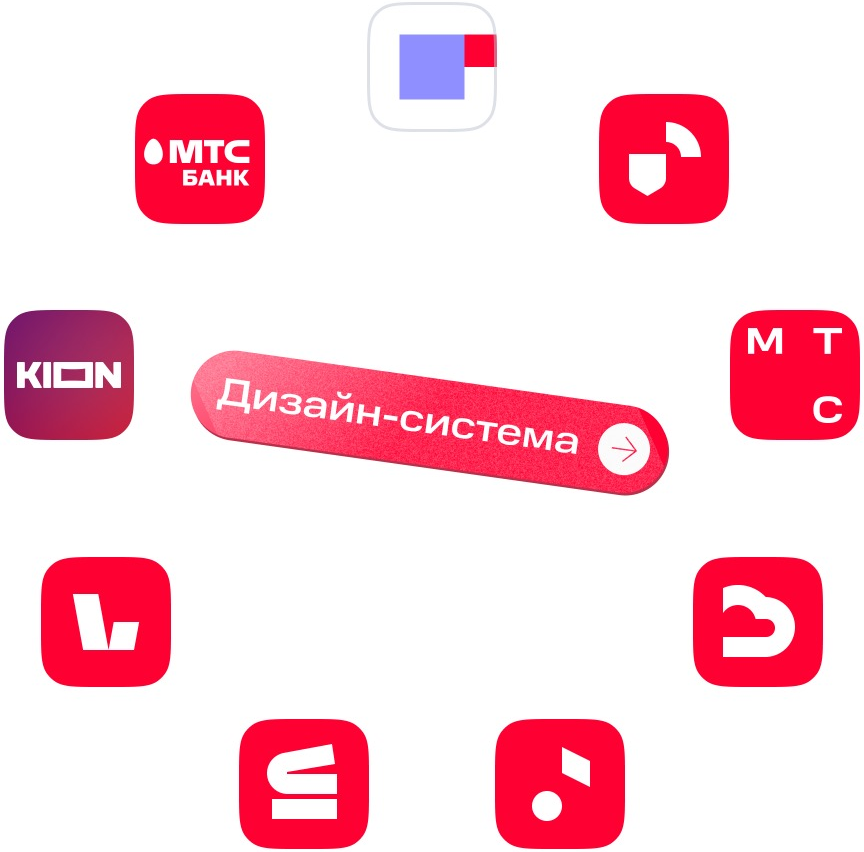
extreme

PROMO SHORT WIDE
Promo Long Wide
H1 32/36 Wide
H2 24/28 Wide
H2 24/28 Comp
H3 20/24 Wide
H3 20/24 Comp
P1 20/28 Regular Comp
P1 20/28 Regular Text
P2 17/24 Bold Comp
P2 17/24 Medium Comp
P2 17/24 Regular Comp
P2 17/24 Regular Text
P3 14/20 Bold Comp
P3 14/20 Medium Comp
P3 14/20 Regular Comp
P3 14/20 Regular Text
P3 14/20 MEDIUM UPPERCASE COMP
P3 14/20 MEDIUM UPPERCASE TEXT
Caption 12/16 Bold Comp
Caption 12/16 Medium Comp
Caption 12/16 Regular Comp
CAPTION 12/16 MEDIUM UPPERCASE COMP
CAPTION 12/16 BOLD UPPERCASE WIDE
CAPTION 10/12 BOLD UPPERCASE WIDE

ПАРАМЕТРЫ ШРИФТА

Итоги и цифры

ЭКОСИСТЕМА МТС

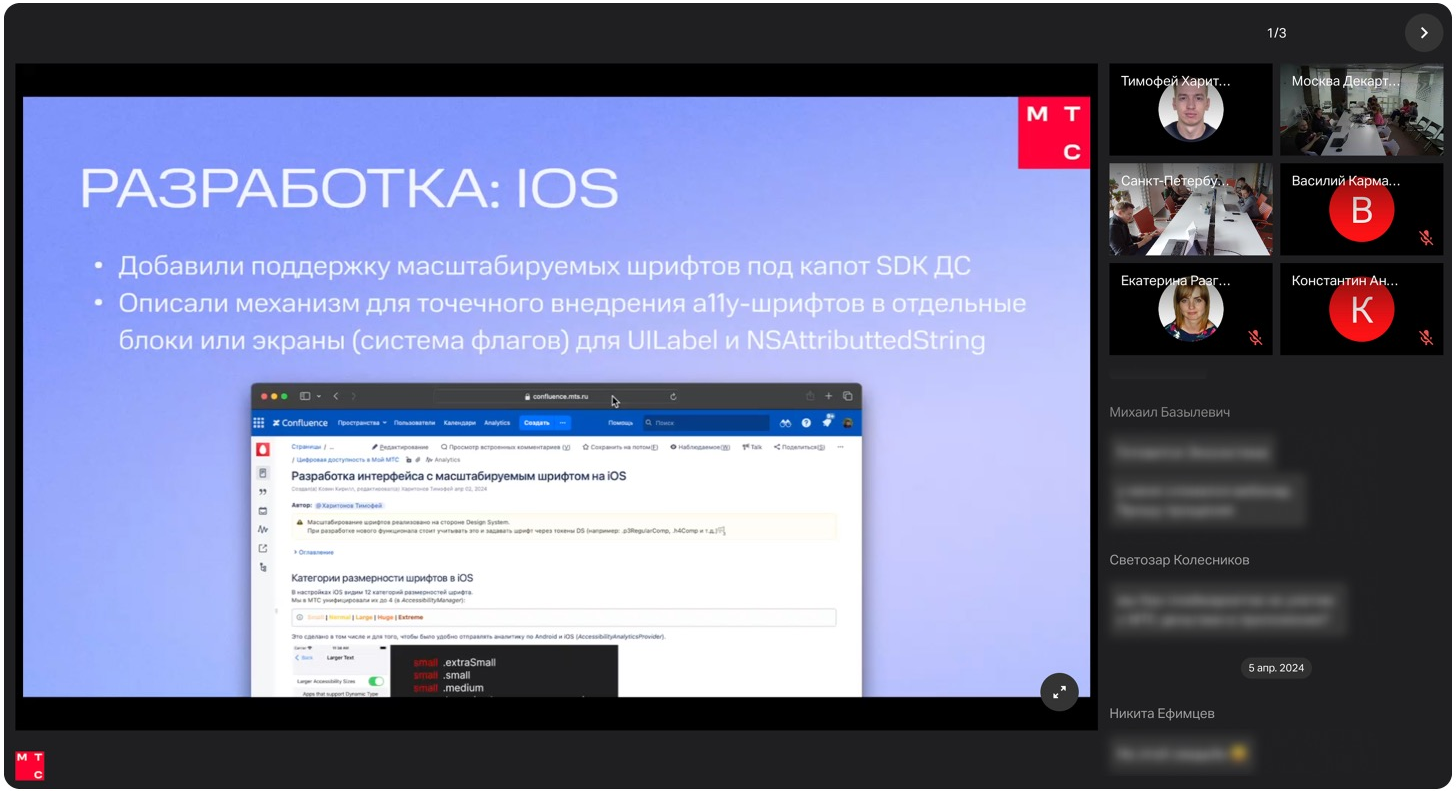


ОЦЕНКА ТРУДОЗАТРАТ

Сложность / Этап	Простой блок Иконка, пара текстовых полей, один лейаут, нет скрытых состояний	Сложный блок Есть несколько состояний, картинки, вариации лейаута	Очень сложный блок Много вложенных элементов, более 5 текстовых полей, несколько настроек
Дизайн	5 мин	10 мин	30 мин
Аналитика	30 мин	60 мин	120 мин
Разработка iOS	60 мин	240 мин	600 мин
Разработка Android	90 мин	300 мин	600 мин
Тестирование	60 мин	240 мин	540 мин
Оценка трудозатрат при первой задаче			
Итого	4 часа (пол дня)	14 часов (2 дня)	31,5 часов (4 дня)

При накопленной экспертизе время разработки сократится на **~30%**
На этапах дизайна, аналитики и тестирования время не изменится

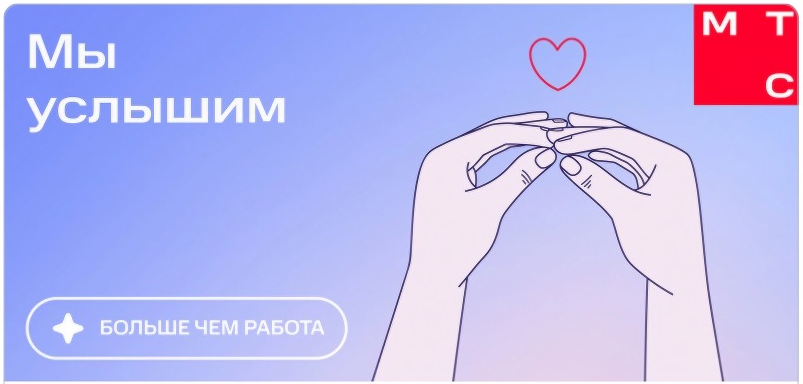
ПРЕЗЕНТАЦИЯ ТЗ КОЛЛЕГАМ В МОЙ МТС



ДОСТИЖЕНИЯ

1. Внедрили решение по A11Y в Дизайн-систему
2. Продумали с дизайнерами процесс взаимодействия и подготовку макетов с A11Y
3. Обучили команды работе с A11Y-шрифтами
4. Наладили межкомандную коммуникацию по 30
5. Адаптировали более 30 блоков, которые используются на множестве экранов

МТС И МЕРОПРИЯТИЯ ПО ИНКЛЮЗИВНОСТИ



Для МТС важна поддержка людей с разными потребностями. Чтобы обратить внимание общества на проблемы слабослышащих, компания организовала Всероссийский инклюзивный фестиваль **«Мы услышим»**.

В рамках фестиваля все желающие смогут больше узнать о цифровых продуктах для людей с инвалидностью и предложить свои инклюзивные проекты. Лучшие идеи рассмотрят команды **MTS StartUp Hub** и **iDA**.

Перейти на сайт

*Рекомендуем смотреть веб-версию

IT

Изучаю

Программа Корп универ

Доступность в iOS-разработке

3 курса

Планы на будущее

ПЛАНЫ НА БУДУЩЕЕ

1. Решить проблему с межстрочным интервалом
2. Уровень доступности AA
3. Внедрить Snapshot тесты
4. Постепенно внедрить масштабирование текста на всех экранах и блоках приложения
5. Углубиться в VoiceOver/TalkBack

СПАСИБО ДИЗАЙНЕРАМ, CX APP И МТС

Мой МТС

СПАСИБО ЗА ВНИМАНИЕ!

Мой МТС

Мой Telegram
для связи

Тестовое
приложения
на GitHub

Мой LinkedIn
для связи

Мой МТС