Building Design System

First baby steps

MobilePay™

Our goals

The goals we want to achieve within next year

Transparent
Consistent
Evolving
Agile

Where we started

The core building block of all design and our apps

Typography

Amount

Title

Headline

Body

Caption

Small

Colours

MP Blue

#5A78FF

MP Blue 80

MP Blue 60

MP Blue 40

MP Blue 20

MP Blue 120

Dark Blue

#3C3246

Dark Blue 80

Dark Blue 60

Dark Blue 40

Dark Blue 20

Dark Blue 120

Violet

#897AFF

Violet 80

Violet 60

Violet 40

Violet 20

Violet 120

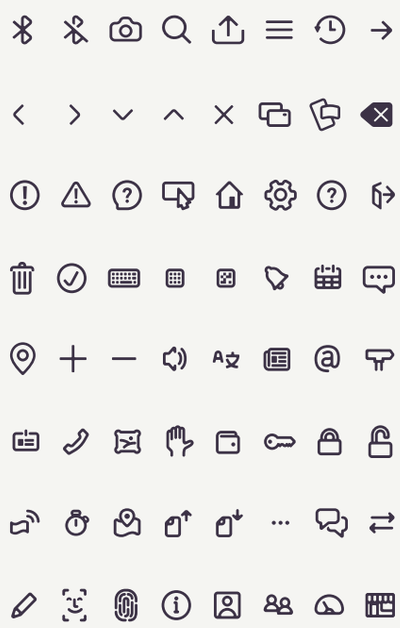
Light Blue

#32E6FF

Turqouise

#00FFD7

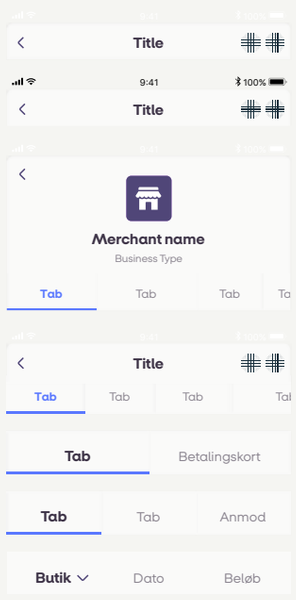
Icons



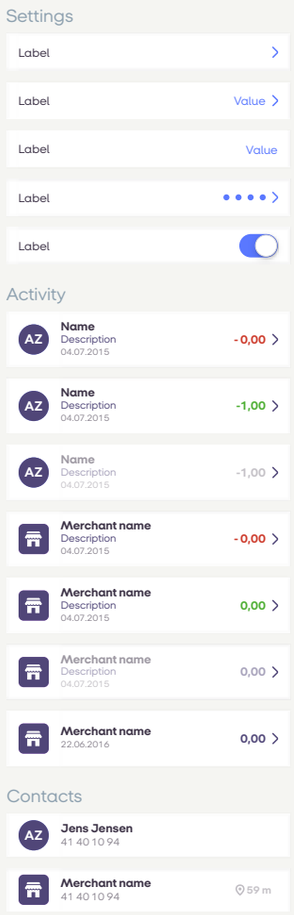
Going forward

More complex elements which are the building blocks for final app screens.

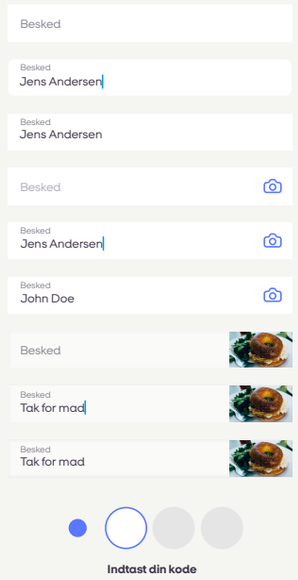
Top Bars



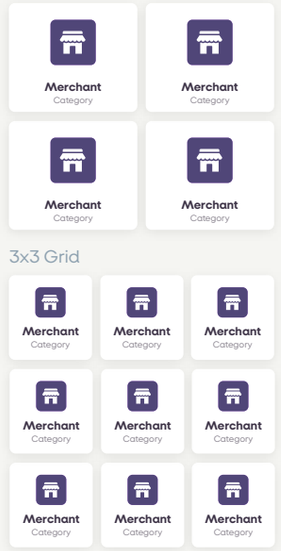
List



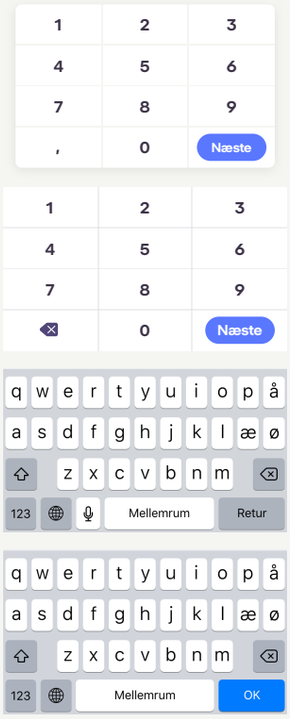
Text Fields



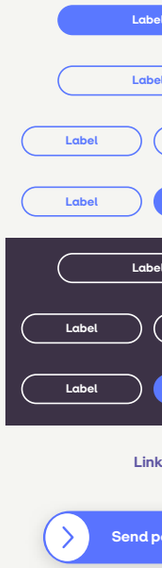
Grid



Keyboards



Buttons



Our Tech Stack

Sketch

Mail tool for design creation

Abstract

Design repository and collaboration tool

Zeplin

Automatic style guide and resource generator

Invision

Prototyping and screen sharing

Design system

iOS implementation

Goals

- Easy to use
- Easy to create new theme
- Multiple theme support

Appearance

```
struct Appearance {
    static let current = Appearance(theme: ThemeLight())

    let animation: Animation
    let cornerRadius: CornerRadius
    let font: Font
    let color: Color
}

// Usage

… = Appearance.current.animation.short
… = Appearance.current.cornerRadius.normal
… = Appearance.current.font.regular.title
… = Appearance.current.font.bold.body
… = Appearance.current.color.white
… = Appearance.current.color.darkBlue40
```

Animation & Corner radius

```
extension Appearance {
    struct Animation {
        let short: TimeInterval
        let medium: TimeInterval
    }
}
```

```
extension Appearance {
    struct CornerRadius {
        let small: CGFloat
        let normal: CGFloat
        let big: CGFloat
    }
}
```

Font & Color

```
extension Appearance {
    struct Font {
        struct Style {
            let title: UIFont
            let body: UIFont
            let small: UIFont
        }

        let regular: Style
        let bold: Style
    }
}

extension Appearance {
    struct Color {
        let white: UIColor

        let darkBlue: UIColor
        let darkBlue80: UIColor
        let darkBlue60: UIColor
        let darkBlue40: UIColor
        let darkBlue20: UIColor
    }
}
```

Init - putting all together

```
extension Appearance {

    private init(theme: AppearanceTheme) {
        self.animation = Animation(
            short: theme.animationShort,
            medium: theme.animationMedium
        )
        self.cornerRadius = CornerRadius(
            small: theme.cornerRadiusSmall,
            normal: theme.cornerRadiusNormal,
            big: theme.cornerRadiusBig
        )
        self.font = Font(
            regular: Appearance.Font.Style(
                title: theme.regularTitleFont,
                body: theme.regularBodyFont,
                small: theme.regularSmallFont
            ),
            bold: Appearance.Font.Style(
                title: theme.boldTitleFont,
                body: theme.boldBodyFont,
                small: theme.boldSmallFont
            )
        )
        self.color = Color(
            white: theme.white,
            darkBlue: theme.darkBlue,
            darkBlue80: theme.darkBlue80,
            darkBlue60: theme.darkBlue60,
            darkBlue40: theme.darkBlue40,
            darkBlue20: theme.darkBlue20
        )
    }
}
```

How theme looks like?

Theme v1

Theme protocol with concrete themes implementation in separate files

+ Themes separated into separate files

– Hard to compare themes because they are in separate files

Theme v2

Theme as enum with themes as values and one extension file with specific fonts, colours, etc. using switch

+ One place where you easily can compare themes

– All themes in one file

Theme v1 example

```
// Theme v1

protocol AppearanceTheme {
    var animationShort: TimeInterval { get }
    var animationMedium: TimeInterval { get }

    var cornerRadiusSmall: CGFloat { get }
    var cornerRadiusNormal: CGFloat { get }
    var cornerRadiusBig: CGFloat { get }

    var regularTitleFont: UIFont { get }
    var regularBodyFont: UIFont { get }
    var regularSmallFont: UIFont { get }

    var boldTitleFont: UIFont { get }
    var boldBodyFont: UIFont { get }
    var boldSmallFont: UIFont { get }

    var white: UIColor { get }

    var darkBlue: UIColor { get }
    var darkBlue80: UIColor { get }
    var darkBlue60: UIColor { get }
    var darkBlue40: UIColor { get }
    var darkBlue20: UIColor { get }
}
```

```
// Theme v1

struct ThemeLight: AppearanceTheme {
    let animationShort: TimeInterval = 0.3
    let animationMedium: TimeInterval = 1

    let cornerRadiusSmall: CGFloat = 4
    let cornerRadiusNormal: CGFloat = 8
    let cornerRadiusBig: CGFloat = 16

    let regularTitleFont: UIFont = .systemFont(ofSize: 16)
    let regularBodyFont: UIFont = .systemFont(ofSize: 12)
    let regularSmallFont: UIFont = .systemFont(ofSize: 8)

    let boldTitleFont: UIFont = .boldSystemFont(ofSize: 16)
    let boldBodyFont: UIFont = .boldSystemFont(ofSize: 12)
    let boldSmallFont: UIFont = .boldSystemFont(ofSize: 8)

    let white: UIColor = .white

    let darkBlue: UIColor = UIColor(red: 60 / 255, green: 50 / 255, blue:
    let darkBlue80: UIColor = UIColor(red: 99 / 255, green: 91 / 255, blue
    let darkBlue60: UIColor = UIColor(red: 138 / 255, green: 132 / 255, b
    let darkBlue40: UIColor = UIColor(red: 177 / 255, green: 173 / 255, b
    let darkBlue20: UIColor = UIColor(red: 216 / 255, green: 214 / 255, b
}
```

```
// Theme v1

protocol AppearanceTheme {
    var animationShort: TimeInterval { get }
    var animationMedium: TimeInterval { get }

    var cornerRadiusSmall: CGFloat { get }
    var cornerRadiusNormal: CGFloat { get }
    var cornerRadiusBig: CGFloat { get }

    var regularTitleFont: UIFont { get }
    var regularBodyFont: UIFont { get }
    var regularSmallFont: UIFont { get }

    var boldTitleFont: UIFont { get }
    var boldBodyFont: UIFont { get }
    var boldSmallFont: UIFont { get }

    var white: UIColor { get }

    var darkBlue: UIColor { get }
    var darkBlue80: UIColor { get }
    var darkBlue60: UIColor { get }
    var darkBlue40: UIColor { get }
    var darkBlue20: UIColor { get }
}
```

Theme v2 example

```
// Theme v2

enum Theme {
    case `default`
    case light
}
```

```swift
// Theme v2

extension Theme {
    var animationShort: TimeInterval {
        switch self {
        case .default: return 0.3
        case .light: return 0.3
        }
    }
    var animationMedium: TimeInterval {
        switch self {
        case .default: return 1
        case .light: return 1
        }
    }
}

extension Theme {
    var cornerRadiusSmall: CGFloat {
        switch self {
        case .default: return 4
        case .light: return 4
        }
    }
    var cornerRadiusNormal: CGFloat {
        switch self {
        case .default: return 8
        case .light: return 8
        }
    }
    var cornerRadiusBig: CGFloat {
        switch self {
        case .default: return 16
        case .light: return 16
        }
    }
}
```

```swift
// Theme v2

extension Theme {
    var regularTitleFont: UIFont {
        switch self {
        case .default: return .systemFont(ofSize: 16)
        case .light: return .systemFont(ofSize: 16)
        }
    }
    var regularBodyFont: UIFont {
        switch self {
        case .default: return .systemFont(ofSize: 12)
        case .light: return .systemFont(ofSize: 12)
        }
    }
    var regularSmallFont: UIFont {
        switch self {
        case .default: return .systemFont(ofSize: 8)
        case .light: return .systemFont(ofSize: 8)
        }
    }
}

extension Theme {
    var boldTitleFont: UIFont {
        switch self {
        case .default: return .boldSystemFont(ofSize: 16)
        case .light: return .boldSystemFont(ofSize: 16)
        }
    }
    var boldBodyFont: UIFont {
        switch self {
        case .default: return .boldSystemFont(ofSize: 12)
        case .light: return .boldSystemFont(ofSize: 12)
        }
    }
    var boldSmallFont: UIFont {
        switch self {
        case .default: return .boldSystemFont(ofSize: 8)
        case .light: return .boldSystemFont(ofSize: 8)
        }
    }
}
```

```swift
// Theme v2

extension Theme {
    var darkBlue: UIColor {
        switch self {
        case .default: return .white
        case .light: return .white
        }
    }
    var darkBlue80: UIColor {
        switch self {
        case .default: return UIColor(red: 60 / 255, green: 50 / 255, blue: 70 / 255, alpha: 1)
        case .light: return UIColor(red: 60 / 255, green: 50 / 255, blue: 70 / 255, alpha: 1)
        }
    }
    var darkBlue60: UIColor {
        switch self {
        case .default: return UIColor(red: 60 / 255, green: 50 / 255, blue: 70 / 255, alpha: 1)
        case .light: return UIColor(red: 60 / 255, green: 50 / 255, blue: 70 / 255, alpha: 1)
        }
    }
    var darkBlue40: UIColor {
        switch self {
        case .default: return UIColor(red: 60 / 255, green: 50 / 255, blue: 70 / 255, alpha: 1)
        case .light: return UIColor(red: 60 / 255, green: 50 / 255, blue: 70 / 255, alpha: 1)
        }
    }
    var darkBlue20: UIColor {
        switch self {
        case .default: return UIColor(red: 60 / 255, green: 50 / 255, blue: 70 / 255, alpha: 1)
        case .light: return UIColor(red: 60 / 255, green: 50 / 255, blue: 70 / 255, alpha: 1)
        }
    }
}
```

Navigation items

```
navigationItem.leftBarButtonItem = .close { [weak self] in
    self?.didTapCloseBarButtonItem()
}
```

```swift
extension UIBarButtonItem {

    typealias TapAction = () -> Void

    static func back(action tapAction: @escaping TapAction) -> UIBarButtonItem {
        return ImageBarButtonItem.make(
            image: .iconNamed("back"),
            accessibilityText: DBLocalizedString("BACK"),
            action: tapAction
        )
    }

    static func close(action tapAction: @escaping TapAction) -> UIBarButtonItem {
        return ImageBarButtonItem.make(
            image: .iconNamed("navigationbar_item_close"),
            accessibilityText: DBLocalizedString("CLOSE"),
            action: tapAction
        )
    }
}
```

```swift
fileprivate final class ImageBarButtonItem: UIBarButtonItem {

    private var tapAction: TapAction?

    static func make(image: UIImage, accessibilityText: String, action tapAction: @escaping TapAction) -> UIBarButton
{
        let item = ImageBarButtonItem(image: image, style: .plain, target: nil, action: nil)
        item.target = item
        item.action = #selector(item.didTap)
        item.accessibilityLabel = accessibilityText
        item.tapAction = tapAction

        return item
    }

    @objc private func didTap() {
        tapAction?()
    }
}
```

Questions?