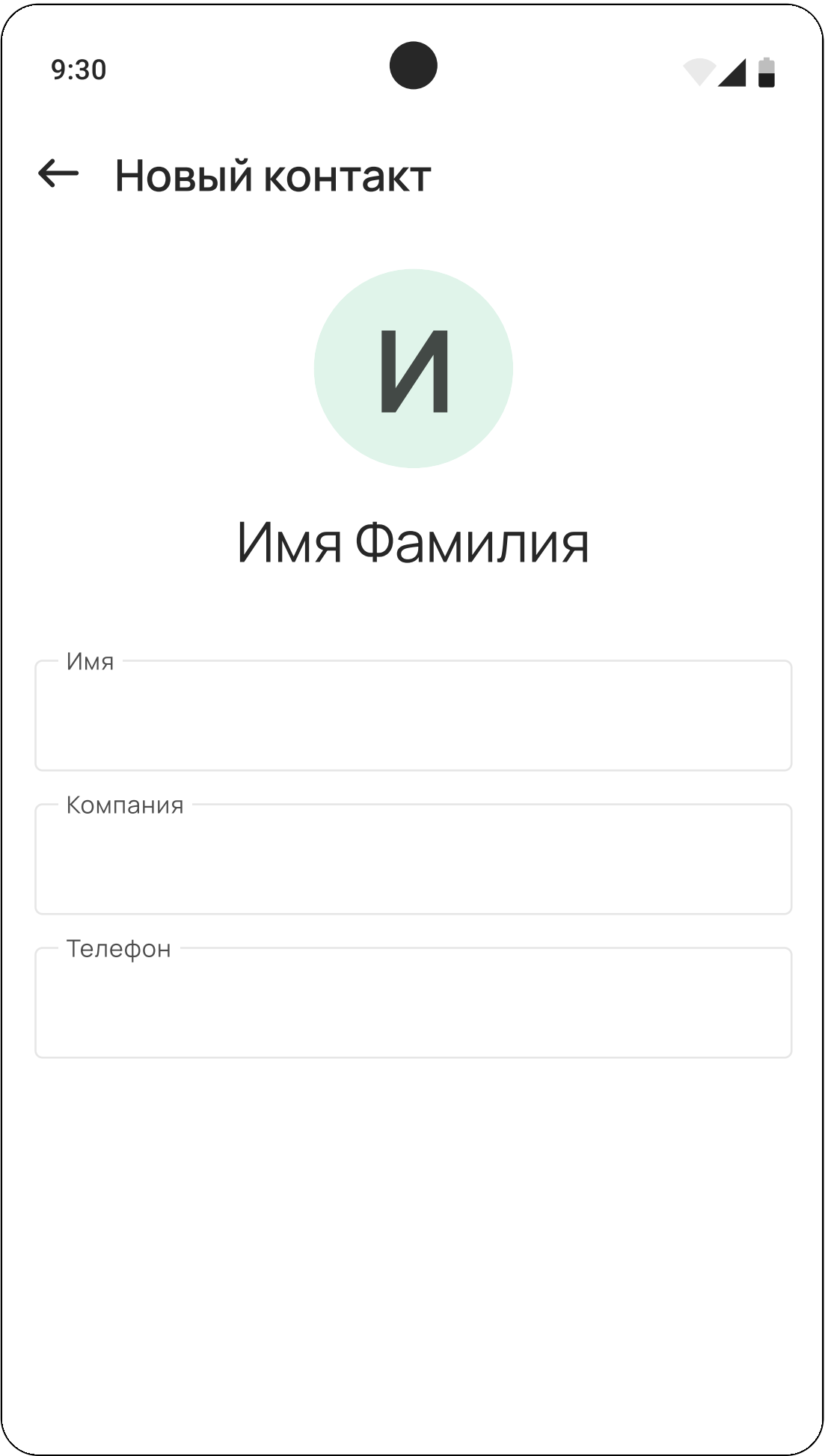


РЕМАРКА

Я никогда до этого не писал что-то вроде документаций, поэтому оно может выглядеть кривенько, однако, коль моей задачей было подробно объяснить почему я сделал все так, а не иначе, тут все будет достаточно подробно, и, возможно, не очень технически. В случаях, когда я сделал то или иное, и не смог объяснить, могу лишь сказать, что я уповал на программистское чутье, содействие кошки и моральную поддержку Степана.

ОСНОВНАЯ ФУНКЦИЯ – CONTACTSCREEN()



Кнопка Сохранить контакт конечно тоже находится в Column, но она и так прижата к низу, так что в блоке она находится для моего удобства

В основной структуре я выделил каждый из элементов, то есть: Бар вроде Header (Новый контакт) Иконку и Имя-Фамилию Блоки полей ввода

А так как они идут друг за другом для удобства я положил их в Column. Опять же для удобства каждый из элементов это отдельная Composable функция, а Блок с полями ввода еще и состоит из отдельных функций-полей (зачем - далее)

```
@Composable
fun ContactScreen() {
    Column(
        modifier = Modifier

    .background(Color.White)
        .fillMaxSize()
    ) {
        Header(title = "Новый
контакт")
        User(content = "Content
string")
        ...

        InputBlock(
            ...
        )

        Spacer(modifier =
Modifier.weight(1f))
        SaveButton(
            ...
        )
    }
}
```

А-ЛЯ HEADER

```
@Composable
fun Header(title: String,
//      onClick: () → Unit)
      isDark: Boolean,
      onToggleTheme: () → Unit
){
    Row(
        verticalAlignment = Alignment.CenterVertically,
        modifier = Modifier
            .fillMaxWidth()
            .padding(
                start = 4.dp,
                end = 4.dp,
                top = 8.dp,
                bottom = 8.dp
            )

    ) {

        Box(
            modifier = Modifier.size(48.dp),
            contentAlignment = Alignment.Center) {

            Image(
                painter = painterResource(id =
R.drawable.arrow_left),
                contentDescription = "back-arrow-button",
                modifier = Modifier,
                colorFilter =
ColorFilter.tint(MaterialTheme.colorScheme.onBackground
)
            //      .clickable { onClick() }
        )
    }
    Spacer(modifier = Modifier.width(4.dp))
    Text(
        text = title,
        style = systemText,
        color = MaterialTheme.colorScheme.onBackground
    )
    Spacer(modifier = Modifier.weight(1f))
    Box(
        modifier = Modifier
            .size(48.dp)
            .clip(CircleShape)
            .clickable { onToggleTheme() },
        contentAlignment = Alignment.Center
    ) {
        val icon = if (isDark) R.drawable.moon else
R.drawable.sun
        Image(
            painter = painterResource(id = icon),
            contentDescription = "theme-changer",
            colorFilter =
ColorFilter.tint(MaterialTheme.colorScheme.onBackground
d)
        )
    }
}
}
```

Хедер мной воспринимался как два элемента, расположенные друг за другом, поэтому - row.

Все padding были взяты из Figma-шаблона, как и шрифты (это, впрочем, относиться ко всем отступам и шрифтам-размерам шрифта в файле)

Единственное исключение из вне - стрелочка. Я взял svg иконку с сайта с иконками для фигмы и поместил ее в код (знакомьтесь - arrow_left)

Она помещена в box по аналогии с тем, как она размещена на примере, т.е. занимая 48*48 квадрат, сама иконка всего 24*24 по центру этого квадрата.

UPD: Так как для переключения на темную тему нужна, собственно, кнопка, header был дополнен еще одной “кнопкой” - нижнем Box с сменяющейся иконкой для смены дневной/ночной темы.

И так как теперь есть темная тема, нужно было убрать статичные Color.Black/Color.White и заменить на цвета из Theme.kt.

Иконка и имя

```
@Composable
fun User(content: String, initial: String) {
    Column (
        modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp, vertical = 8.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Box(
            modifier = Modifier
                .size(100.dp)
                .clip(CircleShape)
                .background(Color(0xFF32BA76).copy(alpha = 0.15f)),
            contentAlignment = Alignment.Center
        ) {
            Text(
                text = initial,
                color =
                    MaterialTheme.colorScheme.onBackground.copy(alpha = 0.7f),
                fontFamily = manropeFontFamily,
                fontSize = 57.sp,
                fontWeight = FontWeight.Bold
            )
        }
        Spacer(modifier = Modifier.height(8.dp))
        Text(
            text = content,
            style = titleLarge,
            color = MaterialTheme.colorScheme.onBackground
        )
    }
}
```

Элементы друг под другом - Column.

Обычный круг с буквой для иконки и текст снизу.

Из интересного - текст и буква на иконке синхронизирована со вводом в поле Имя. Текст копирует имя, буква - первую букву соответственно (с учетом того, что пробелы учитывать не надо и если ничего не введено, показываем пользователю загадочный “?”).

БЛОК С ПОЛЯМИ ВВОДА

```
@Composable
fun InputBlock(
    name: String,
    onNameChange: (String) → Unit,
    company: String,
    onCompanyChange: (String) → Unit,
    phone: String,
    onPhoneChange: (String) → Unit
) {
    Column(
        modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp, vertical = 16.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        InputField(inform = "Имя", value = name,
            onValueChange = onNameChange)
        Spacer(modifier = Modifier.height(16.dp))
        InputField(inform = "Компания", value = company,
            onValueChange = onCompanyChange)
        Spacer(modifier = Modifier.height(16.dp))
        InputField(inform = "Телефон", value = phone,
            onValueChange = onPhoneChange, keyboardType =
                KeyboardType.Phone)
    }
}
```

Так как опираясь на опыт программирования, а именно - “если что-то повторяется, оберни это в отдельную функцию, чтобы потом не переписывать 800 разных строк в разных файлах при первой правке” - поля ввода были сделаны отдельными composable-функциями, а блок с полями ввода нужен для их правильного размещения.

ПОЛЕ ВВОДА

```
@Composable
fun InputField(
    inform: String,
    value: String,
    onChange: (String) → Unit,
    keyboardType: KeyboardType = KeyboardType.Text
) {
    Box(
        modifier = Modifier
            .fillMaxWidth()
            .height(56.dp)
    ) {
        OutlinedTextField(
            value = value,
            onChange = onChange,
            modifier = Modifier
                .fillMaxSize(),
            textStyle = TextStyle(
                fontSize = 16.sp,
                color = MaterialTheme.colorScheme.onBackground
            ),
            keyboardOptions = KeyboardOptions.Default.copy(
                keyboardType = keyboardType
            ),

            colors = OutlinedTextFieldDefaults.colors(
                unfocusedBorderColor =
                    MaterialTheme.colorScheme.onBackground.copy(alpha =
                        0.15f) // ← цвет при отсутствии фокуса
            )
        )

        Text(
            modifier = Modifier
                .offset(x = 12.dp, y = (-8).dp)

            .background(MaterialTheme.colorScheme.background)
                .padding(
                    horizontal = 4.dp
                )
            ,

            text=inform,
            color =
                MaterialTheme.colorScheme.onBackground.copy(alpha =
                    0.7f),
            fontFamily = manropeFontFamily,
            fontSize = 12.sp,
            letterSpacing = 0.4.sp,

        )
    }
}
```

Ну, это кнопка. Я полагаю..?

Функция `onClick` для сохранения информации в `Log`; загадочные дополнительные отступы снизу в `Figma` (там просто почему-то кнопка еще выше отступа. Скорее всего выставлено вручную, но мы же делаем макет по шаблону, значит по шаблону)

Так как цвет в обоих темах один, тут он задается фиксированным.

Высота была взята из `Figma` - 56px

Не ясно было, как максимально правильно расположить текст, идущий поверх контура, поэтому он зафиксирован “грубо и топорно” - обычным перемещением (`.offset(x = 12.dp, y = (-8).dp)`)

Ну и добавление `keyboardType` - так как у нас есть поле для ввода телефона и в теории может быть поле для ввода еще чего-то, отличного от текста, поле было по умолчанию определено как текст, но по надобности может быть переназначена на любой из типов.

КНОПКА. ПРОСТО КНОПКА

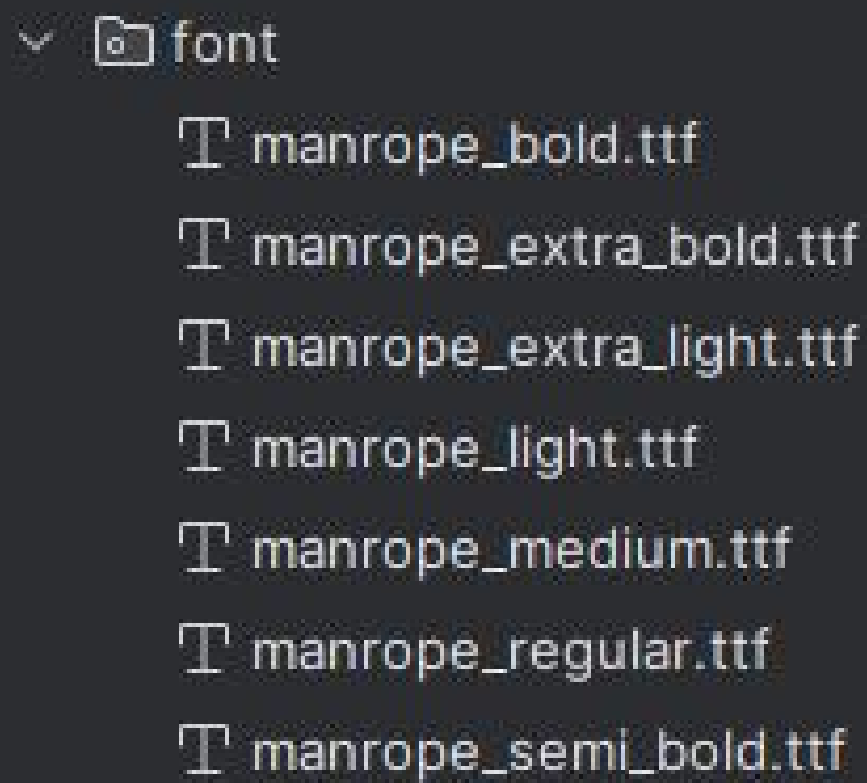
```
@Composable
fun SaveButton(
    name: String,
    company: String,
    phone: String
) {
    Button(
        onClick = {
            Log.d("ContactForm", "Имя: $name, Компания: $company, Телефон: $phone")
        },
        modifier = Modifier
            .fillMaxWidth()
            .padding(16.dp, bottom = (16+8).dp) // ← загадочные 8px в figma
            .height(56.dp),
        shape = RoundedCornerShape(100.dp),
        colors = ButtonDefaults.buttonColors(
            containerColor = Color(0xFFFFFC107),
            contentColor = Color.Black
        )
    ) {
        Text(
            text = "Добавить контакт",
            fontSize = 16.sp,
            fontFamily = manropeFontFamily
        )
    }
}
```


ДОПОЛНИТЕЛЬНОЕ

```
private val DarkColorScheme = darkColorScheme(
    background = Color.Black,
    onBackground = Color.White
)

private val LightColorScheme = lightColorScheme(
    background = Color.White,
    onBackground = Color.Black
)
```

Тут буквально пара строчек для того, чтобы темная/светлая тема работала.



```
font
├── manrope_bold.ttf
├── manrope_extra_bold.ttf
├── manrope_extra_light.ttf
├── manrope_light.ttf
├── manrope_medium.ttf
├── manrope_regular.ttf
└── manrope_semi_bold.ttf
```

```
var systemText = TextStyle(
    fontFamily = manropeFontFamily,
    fontWeight = FontWeight.SemiBold,
    fontSize = 22.sp,
)

var titleLarge = TextStyle (
    fontFamily = manropeFontFamily,
    fontWeight = FontWeight.Normal,
    fontSize = 28.sp
)
```

Шрифты добавить было элементарно, однако я углядел интересную вещь - `TextStyle`ы` и я возможно немного не понимаю, как это работает, но я решил попробовать определить пару стилей.

Из минусов:

- было потрачено время на чтение `metanit`а`

Из плюсов:

- это было интересно
- это сэкономило +-10 строк в коде основного файла
- это в теории удобно править (как с функциями - указал все в одном месте и только тут и редактируешь)