Android Studio - Zegar cyfrowy

Ogólne

Aplikacja prezentuje statyczne wyświetlanie czasu oraz dynamiczną wersję zegara cyfrowego.

Uwaga: Nigdy nie wolno kopiować kodu z PDF-a, gdyż zawiera on niewidoczne znaki końca linii, twarde odstępy, odmienne cudzysłowy, odmienne apostrofy, odmienne myślniki. To godziny dodatkowej pracy na wykrywanie błędów i ich poprawianie. Kod należy przepisać ze zrozumieniem.

Kod - wersja podstawowa

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ZegarTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    WszystkieElementy()
                }
            }
        }
    }
}


@Composable
fun KolorowyTekstZegarStatyczny1() {
    val sformatowanaData =
        LocalDateTime.now().format(DateTimeFormatter.ofPattern("HH:mm, dd-MM-yyyy"))
    val tekst = "Czas pochodzący od zmiennej: $sformatowanaData."
    Text(tekst)
}

@Composable
fun KolorowyTekstZegarStatyczny2() {
    val czas = getDateTimeInstance().format(Date()) // Bez możliwości formatowania daty
    val tekst = "Drugi sposób: $czas."
     Text(tekst)
}
```

```
@Composable
fun KolorowyTekstZegarStatyczny3() {
    val godzinaSimple = SimpleDateFormat("HH:mm").format(Date())
    val tekst = "Czwarty sposób (simple): $godzinaSimple."
    Text(tekst)
}


@Composable
fun KolorowyTekstZegarDynamiczny1() {
    var obecnyCzasDynamiczny1 by remember {
        mutableStateOf(
            SimpleDateFormat(
                "HH:mm:ss",
                Locale.getDefault()
            ).format(Date())
        )
    }
    LaunchedEffect(Unit) {
        while (true) {
            obecnyCzasDynamiczny1 = SimpleDateFormat("HH:mm:ss",
            Locale.getDefault()).format(Date())
            delay(1000)
        }
    }
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text("W stylu JetPack Compose:", textAlign = TextAlign.Center)
        Spacer(modifier = Modifier.height(30.dp))
        Text(
            obecnyCzasDynamiczny1,
            textAlign = TextAlign.Center,
            fontSize = 60.sp,
            color = Color.Red
        )
    }
}


@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun WszystkieElementy() {
    Column {
        CenterAlignedTopAppBar(
```

```
            title = { Text("Zegary statyczne i dynamiczne", fontWeight = FontWeight.ExtraBold) },
            colors = TopAppBarDefaults.topAppBarColors(
                containerColor = Color(47, 79, 79, 255),
                titleContentColor = Color.White
            )
        )

        Spacer(modifier = Modifier.height(40.dp))
        Text(
            "===== Zegary statyczne =====",
            fontWeight = FontWeight.Bold, modifier = Modifier.align(
                Alignment.CenterHorizontally
            ), color = Color.Cyan, fontSize = 20.sp
        )
// Zegar 1
        Spacer(modifier = Modifier.height(30.dp))
        KolorowyTekstZegarStatyczny1()
// Zegar 2
        Spacer(modifier = Modifier.height(30.dp))
        KolorowyTekstZegarStatyczny2()
// Zegar 3
        Spacer(modifier = Modifier.height(30.dp))
        KolorowyTekstZegarStatyczny3()

// ================ DYNAMICZNE ======================
        Spacer(modifier = Modifier.height(100.dp))
        Text(
            "===== Zegar dynamiczny =====",
            fontWeight = FontWeight.Bold,
            modifier = Modifier.align(
                Alignment.CenterHorizontally
            ),
            color = Color.Cyan,
            fontSize = 20.sp
        )
// Zegar 4
        Spacer(modifier = Modifier.height(30.dp))
        KolorowyTekstZegarDynamiczny1()
    }
}
```

Kod - wersja rozbudowana

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ZegarTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    WszystkieElementy()
                }
            }
        }
    }
}
```

```
@Composable
fun KolorowyTekstZegarStatyczny1() {
    val sformatowanaData =
        LocalDateTime.now().format(DateTimeFormatter.ofPattern("HH:mm, dd-MM-yyyy"))
    val tekst = "Czas pochodzący od zmiennej: $sformatowanaData."
    // Chcemy, aby tylko wyświetlana data/godzina była koloru czerwonego:
    val sformatowanyTekst = buildAnnotatedString {
        append(tekst)
        addStyle(
            style = SpanStyle(color = Color.Red),
            start = tekst.indexOf(sformatowanaData), // Tylko data na czerwono
            end = tekst.indexOf(sformatowanaData) + sformatowanaData.length
        )
    }
    Text(text = sformatowanyTekst)
}

@Composable
fun KolorowyTekstZegarStatyczny2() {
    val czas = getDateTimeInstance().format(Date()) // Bez możliwości formatowania daty
    val tekst = "Drugi sposób: $czas."
    val sformatowanyTekst = buildAnnotatedString {
        append(tekst)
        addStyle(
            style = SpanStyle(color = Color.Red),
            start = tekst.indexOf(czas), // Tylko czas na czerwono
            end = tekst.indexOf(czas) + czas.length
        )
```

```
    }
    Text(text = sformatowanyTekst)
}

@Composable
fun KolorowyTekstZegarStatyczny3() {
    val czasJava = Calendar.getInstance()
    // W poniższym przykładzie, funkcja rozszerzenia padStart uzupełnia początek
    // wyrażenia znakiem 0, tak aby zawsze były dwa znaki w godzinie i minucie:
    val godzina = czasJava.get(Calendar.HOUR_OF_DAY).toString().padStart(2, '0')
    val minuta = czasJava.get(Calendar.MINUTE).toString().padStart(2, '0')
    val tekst = "Trzeci sposób (tzw. Java Style): $godzina:$minuta."
    val sformatowanyTekst = buildAnnotatedString {
        append(tekst)
        addStyle(
            style = SpanStyle(color = Color.Red),
            start = tekst.indexOf("$godzina:$minuta"), // Godzina i minuta na czerwono
            end = tekst.indexOf("$godzina:$minuta") + "$godzina:$minuta".length
        )
    }
    Text(text = sformatowanyTekst)
}


@Composable
fun KolorowyTekstZegarStatyczny4() {
    val godzinaSimple = SimpleDateFormat("HH:mm").format(Date())
    val tekst = "Czwarty sposób (simple): $godzinaSimple."
    val sformatowanyTekst = buildAnnotatedString {
        append(tekst)
        addStyle(
            style = SpanStyle(color = Color.Red),
            start = tekst.indexOf(godzinaSimple), // Tylko czas na czerwono
            end = tekst.indexOf(godzinaSimple) + godzinaSimple.length
        )
    }
    Text(text = sformatowanyTekst)
}


@Composable
fun KolorowyTekstZegarDynamiczny1() {
    var obecnyCzasDynamiczny1 by remember {
        mutableStateOf(
            SimpleDateFormat(
                "HH:mm:ss",
                Locale.getDefault()
            ).format(Date())
```

```
        )
    }
    // Operacje asynchroniczne, wykonywane poza wątkiem głównym aplikacji,
    // gdyż mogą potrzebować więcej czasu (np. pobieranie informacji z Internetu).
    // Są wątkiem pobocznym (oderwanym, niezależnym, w tle) od głównego wątku:
    LaunchedEffect(Unit) {
        // Chcemy tu mieć wieczną pętlę (odczytuje bieżący czas co sekundę):
        while (true) {
            obecnyCzasDynamiczny1 = SimpleDateFormat("HH:mm:ss",
            Locale.getDefault()).format(Date())
            delay(1000)
        }
    }
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text("W stylu JetPack Compose:", textAlign = TextAlign.Center)
        Spacer(modifier = Modifier.height(30.dp))
        Text(
            obecnyCzasDynamiczny1,
            textAlign = TextAlign.Center,
            fontSize = 60.sp,
            color = Color.Red
        )
    }
}


@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun WszystkieElementy() {
    Column {
        CenterAlignedTopAppBar(
            title = { Text("Zegary statyczne i dynamiczne", fontWeight = FontWeight.ExtraBold) },
            colors = TopAppBarDefaults.topAppBarColors(
                containerColor = Color(47, 79, 79, 255),
                titleContentColor = Color.White
            )
        )

        Spacer(modifier = Modifier.height(40.dp))
        Text(
            "===== Zegary statyczne =====", fontWeight = FontWeight.Bold, modifier =
            Modifier.align(
                Alignment.CenterHorizontally
```

```
            ), color = Color.Cyan, fontSize = 20.sp
        )
// Zegar 1
        Spacer(modifier = Modifier.height(30.dp))
        KolorowyTekstZegarStatyczny1()
// Zegar 2
        Spacer(modifier = Modifier.height(30.dp))
        KolorowyTekstZegarStatyczny2()
// Zegar 3
        Spacer(modifier = Modifier.height(30.dp))
        KolorowyTekstZegarStatyczny3()
// Zegar 4
        Spacer(modifier = Modifier.height(30.dp))
        KolorowyTekstZegarStatyczny4()

// ================ DYNAMICZNE ======================
        Spacer(modifier = Modifier.height(100.dp))
        Text(
            "===== Zegar dynamiczny =====",
            fontWeight = FontWeight.Bold,
            modifier = Modifier.align(
                Alignment.CenterHorizontally
            ),
            color = Color.Cyan,
            fontSize = 20.sp
        )
// Zegar 5
        Spacer(modifier = Modifier.height(30.dp))
        KolorowyTekstZegarDynamiczny1()
    }
}
```

Ostatnia aktualizacja: 11 marca 2024.