МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа № 2**

*по дисциплине «Программирование мобильных устройств»*

Выполнил студен группы ПИбд-32

Преснякова В. В.

Проверил доцент кафедры

«Информационные системы»

Филиппов А.А.

Ульяновск, 2023

**Задание** Разработка экранных форм приложения и настройка навигации.

Необходимо:

1. На основе макетов, полученных в рамках ЛР № 1, разработать экранные формы приложения. При разработке экранных форм можно использовать фрагменты и/или композицию (compose).

2. Настроить навигацию между экранными формами приложения.

3. Разработать необходимые классы-сущности и создать на их основе тестовые наборы данных.

4. Использовать тестовые данные для демонстрации работы приложения.

5. Отчет и изменения проекта загрузить в репозиторий по адресу <http://student.git.athene.tech>

**Решение**

Настроим панель навигации для проекта

Создадим класс для хранения элементов навигации

sealed class NavItem(val route: String, val icon: ImageVector?){  
object Home : NavItem("home", Icons.Default.*Home*)  
object MyOrder : NavItem("myorder", null)  
object Profile : NavItem("profile", Icons.Default.*Person*)  
object SignIn : NavItem("login", null)  
object SignUp : NavItem("signup", null)  
object HotelInfo : NavItem("HotelInfo/{hotelItem}", null)  
object Booking : NavItem("booking/{hotelItem}", null)  
object Person : NavItem("person", null)  
object AdminPanel : NavItem("admin", Icons.Default.*Build*)  
object AddPanel : NavItem("add", null)  
object ChangePanel : NavItem("change", null)  
object ChangeHotel : NavItem("changeHotel/{hotelItem}", null)

}

Далее для навигации настроим NavController который будет отвечать, за то на какую функцию отрисовки мы хотим вызвать

@Composable  
fun NavController(navController: NavHostController){  
 NavHost(  
 navController = navController,  
 startDestination = NavItem.Home.route  
) **{** *composable*(NavItem.HotelInfo.route) **{** //  
 backStackEntry **->** val hotelItemString = backStackEntry.arguments?.getString("hotelItem")  
 val hotelItem = Gson().fromJson(hotelItemString, Hotel::class.*java*)  
 hotelItem?.*let* **{** HotelInfo(**it**, navController)  
 **}  
 }** *composable*(NavItem.Home.route) **{** HomeScreen(navController)  
 **}** *composable*(NavItem.MyOrder.route)**{** MyOrderScreen()  
 **}** *composable*(NavItem.Profile.route) **{** ProfileScreen(navController)  
 **}** *composable*(NavItem.SignIn.route)**{** LoginScreen(navController)  
 **}** *composable*(NavItem.SignUp.route)**{** SignUpScreen(navController)  
 **}** *composable*(NavItem.Booking.route) **{** backStackEntry **->** val hotelItemString = backStackEntry.arguments?.getString("hotelItem")  
 val hotelItem = Gson().fromJson(hotelItemString, Hotel::class.*java*)  
 hotelItem?.*let* **{** BookingScreen(**it**, navController)  
 **}  
 }** *composable*(NavItem.ChangeHotel.route) **{** backStackEntry **->** val hotelItemString = backStackEntry.arguments?.getString("hotelItem")  
 val hotelItem = Gson().fromJson(hotelItemString, Hotel::class.*java*)  
 hotelItem?.*let* **{** ChangeHotel(**it**, onBackClick = **{** navController.navigateUp() **}**)  
 **}  
 }** *composable*(NavItem.Person.route) **{** Person(navController)  
 **}** *composable*(NavItem.AdminPanel.route)**{** AdminPanel(navController)  
 **}** *composable*(NavItem.AddPanel.route)**{** AddPanel(navController)  
 **}** *composable*(NavItem.ChangePanel.route)**{** ChangePanel(navController)  
 **}**

**}**}

И сама функция отрисовки нижней панели навигации

@SuppressLint("UnusedMaterialScaffoldPaddingParameter")  
@Composable  
fun Navigate(){  
 val navController = rememberNavController()  
 val listItem = *listOf*(  
 NavItem.Home,  
 NavItem.Like,  
 NavItem.Order,  
 NavItem.Profile,  
 NavItem.AdminPanel,  
 )  
  
 Scaffold(bottomBar = **{** BottomNavigation(  
 backgroundColor = Color.White  
 ) **{** val navBackStackEntry = navController.currentBackStackEntryAsState()  
 val currentState = navBackStackEntry.value  
  
 listItem.*forEach* **{** it **->** val isSelected = currentState?.destination?.route == it.route  
  
 BottomNavigationItem(  
 selected = isSelected,  
 onClick = **{** if(!isSelected){  
 navController.graph.startDestinationRoute?.*let* **{** navController.popBackStack(**it**, inclusive = true)  
 **}** navController.navigate(it.route)**{** launchSingleTop  
 **}** }  
 navController.navigate(it.route)  
  
 **}**,  
 icon = **{** val iconModifier = if (isSelected) {  
 Modifier  
 .*background*(color = colorResource(id = R.color.*figma\_blue*), shape = *CircleShape*)  
 .*padding*(8.*dp*)  
 } else {  
 Modifier  
 }  
  
 it.icon?.*let* **{** it1 **->** Icon(  
 imageVector = it1,  
 contentDescription = null,  
 modifier = iconModifier.then(Modifier.*size*(24.*dp*))  
 )  
 **}  
 }** )  
 **}  
 }  
 }**) **{** NavController(navController = navController)  
 **}**}

Нижняя панель выглядит следующим образом



Чтобы это все работало и находилось на экране, вызовем их в классе MainActivity

class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *setContent* **{** MainContent()  
 **}** }  
}  
  
@Composable  
fun MainContent() {  
 Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 ) **{**  
 Navigate()  
 **}**}

Далее перейдем к созданию самих экранных форм согласно макету

Для разбиения отдельных компонентов будет выносить их в разные функции

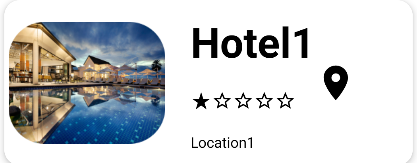
**Главная страница**

Начнем сверху вниз и создадим поле для ввода текста для поиска по каталогу

@Composable  
fun SearchField(  
 modifier: Modifier = Modifier,  
 onSearch: (String) -> Unit  
) {  
 var searchText by remember **{** *mutableStateOf*("") **}** Box(  
 modifier = modifier  
 .*fillMaxWidth*()  
 .*background*(colorResource(id = R.color.*figma*))  
 .*padding*(horizontal = 16.*dp*, vertical = 8.*dp*)  
 ) **{** Row(  
 modifier = Modifier  
 .*fillMaxWidth*(),  
 verticalAlignment = Alignment.CenterVertically,  
 content = **{** BasicTextField(  
 value = searchText,  
 onValueChange = **{** searchText = **it** onSearch(**it**)  
 **}**,  
 textStyle = TextStyle(color = Color.Black),  
 modifier = Modifier  
 .*weight*(1f)  
 )  
  
 SearchIcon(  
 imageVector = Icons.Default.*Search*,  
 contentDescription = "Search Icon",  
 modifier = Modifier.*size*(24.*dp*)  
 )  
 **}** )  
 **}**}

Карточка для отеля

@Composable  
@Composable  
fun HotelCard (hotel: Hotel, navController: NavHostController){  
 androidx.compose.material.Card(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(10.*dp*)  
 .*clickable* **{** val hotelItemString = Gson().toJson(hotel)  
 navController.navigate("HotelInfo/${hotelItemString}")  
 **}**,  
 shape = *RoundedCornerShape*(15.*dp*),  
 elevation = 5.*dp* ) **{** Box(  
 modifier = Modifier.*background*(Color.White)  
 ) **{** Row(  
 modifier = Modifier  
 //.background(Color.Yellow)  
 .*fillMaxWidth*(),  
 verticalAlignment = Alignment.CenterVertically  
  
 ) **{** Image(  
 painter = painterResource(id = hotel.img),  
 contentDescription = "hotel",  
 contentScale = ContentScale.Fit,  
 modifier = Modifier  
 .*padding*(4.*dp*)  
 .*size*(150.*dp*)  
  
 )  
 Column(  
 modifier = Modifier  
 //.background(Color.Red)  
 .*padding*(start = 20.*dp*),  
 //horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.spacedBy(20.*dp*)  
 ) **{** Text(text = hotel.name, fontSize = 40.*sp*, fontWeight = FontWeight.Bold)  
 Row() **{** for (i in 1..hotel.stars){  
 Image(  
 painter = painterResource(id = R.drawable.*star\_rate*),  
 contentDescription = "star",  
 modifier = Modifier  
 .*size*(20.*dp*)  
 )  
 }  
 for (i in 1.. 5 - hotel.stars){  
 Image(  
 painter = painterResource(id = R.drawable.*star\_outline*),  
 contentDescription = "star",  
 modifier = Modifier.*size*(20.*dp*)  
 )  
 }  
 **}** Text(text = hotel.location)  
 **}** Image(  
 imageVector = Icons.Filled.*LocationOn*,  
 contentDescription = "location",  
 modifier = Modifier.*size*(40.*dp*)  
 )  
 **}  
 }  
  
 }**}

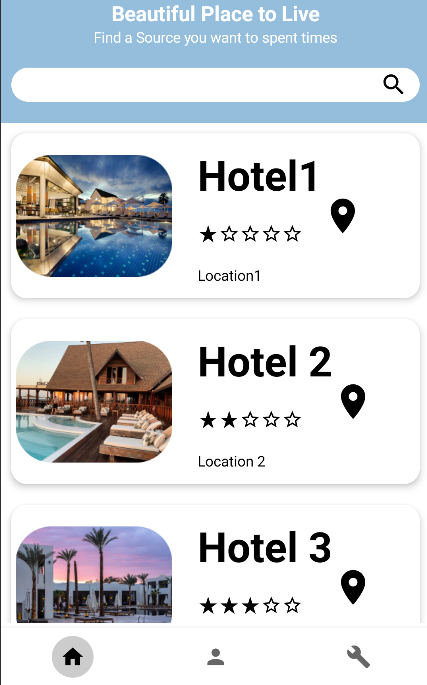


Теперь будем вызывать все эти функции в одной главной для страницы

@Composable  
@Composable  
fun HomeScreen(navController: NavHostController) {  
   
 Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*background*(Color.White)  
 //.padding(bottom = 60.dp)  
 ) **{** Box(modifier = Modifier  
 .*background*(colorResource(id = R.color.*figma\_blue*))  
 .*fillMaxHeight*(0.18f)  
 )**{** Column(  
 modifier = Modifier  
 .*fillMaxHeight*(),  
 horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.SpaceEvenly  
 )**{** Text(text= "Beautiful Place to Live", fontSize = 20.*sp*, fontWeight = FontWeight.Bold,  
 color = Color.White)  
 Text(text="Find a Source you want to spent times", color = Color.White)  
 SearchField(  
 modifier = Modifier  
 .*padding*(horizontal = 10.*dp*, vertical = 20.*dp*),  
  
 ) **{** searchText **->** // Обработка введенного текста поиска  
 **}  
 }  
 }** Column (  
 modifier = Modifier  
 //.verticalScroll(rememberScrollState())  
 .*padding*(bottom = 60.*dp*)  
  
 )**{** LazyVerticalGrid(  
 columns = GridCells.Fixed(1)  
 ) **{**   
 HotelCard(hotel, navController)  
 }  
 **}  
 }  
  
 }  
 }**}

}

В целом страница будет выглядеть так



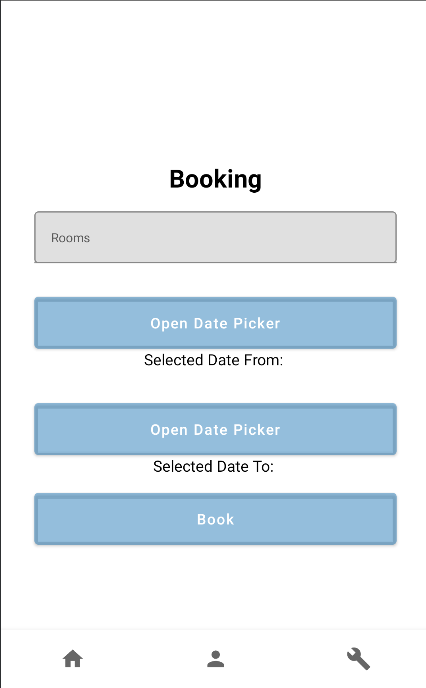
**Страница с отелем**

@Composable  
fun HotelInfo(hotel: Hotel, navController: NavHostController) {  
 Log.d("MyLog", hotel.toString())  
  
 Column (  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*fillMaxHeight*()  
 .*background*(Color.White)  
 .*verticalScroll*(rememberScrollState()),  
 verticalArrangement = Arrangement.SpaceBetween,  
  
 )**{** Image(  
 painter = painterResource(id = hotel.img),  
 contentDescription = "hotel",  
 contentScale = ContentScale.FillWidth,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 )  
 Box(  
 modifier = Modifier.*background*(Color.White)  
 )**{** Row (  
 modifier = Modifier  
 .*fillMaxWidth*(),  
  
 horizontalArrangement = Arrangement.SpaceEvenly  
 )**{** Text(text = hotel.name)  
 // stars  
 **}  
 }** Divider(color = Color.Black, thickness = 1.*dp*)  
 Box(  
 modifier = Modifier.*background*(Color.White)  
 )**{** Row (  
 modifier = Modifier  
 .*fillMaxWidth*(),  
 horizontalArrangement = Arrangement.SpaceEvenly  
 )**{** Text(text = "1 room | 1 Night")  
 Text(text = "Rs. 4000")  
 **}  
 }** Divider(color = Color.Black, thickness = 1.*dp*)  
 Box(  
 modifier = Modifier  
 .*background*(Color.White)  
 )**{** Column (  
 modifier = Modifier  
 .*fillMaxWidth*()  
 ,  
 horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.SpaceEvenly  
 )**{** Text(text = "Location")  
 Text(text = hotel.location)  
 **}  
 }** Divider(color = Color.Black, thickness = 1.*dp*)  
 Box(  
 modifier = Modifier  
 .*background*(Color.White)  
 )**{** Column (  
 modifier = Modifier  
 .*fillMaxWidth*()  
 ,  
 horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.SpaceEvenly  
 )**{** Text(text = "Amenties")  
 Text(text = "wifi")  
 **}  
 }** Divider(color = Color.Black, thickness = 1.*dp*)  
  
 Box(  
 modifier = Modifier  
 .*background*(Color.White)  
 )**{** Column (  
 modifier = Modifier  
 .*fillMaxWidth*()  
 ,  
 horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.SpaceEvenly  
 )**{** Text(text = "Info")  
 Text(text = "inf")  
 **}  
 }** Row(  
 modifier = Modifier  
 .*padding*(bottom = 60.*dp*),  
 verticalAlignment = Alignment.Bottom  
 )**{** Button(  
 colors = ButtonDefaults.buttonColors(  
 backgroundColor = (colorResource(id = R.color.*figma\_blue*)),  
 contentColor = Color.White  
 ),  
 onClick = **{** val hotelItemString = Gson().toJson(hotel)  
 navController.navigate("booking/${hotelItemString}")  
 **}**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(16.*dp*, 16.*dp*, 16.*dp*, 5.*dp*)  
 .*height*(50.*dp*)  
 ) **{** Text("Select Room")  
 **}  
 }  
 }**}



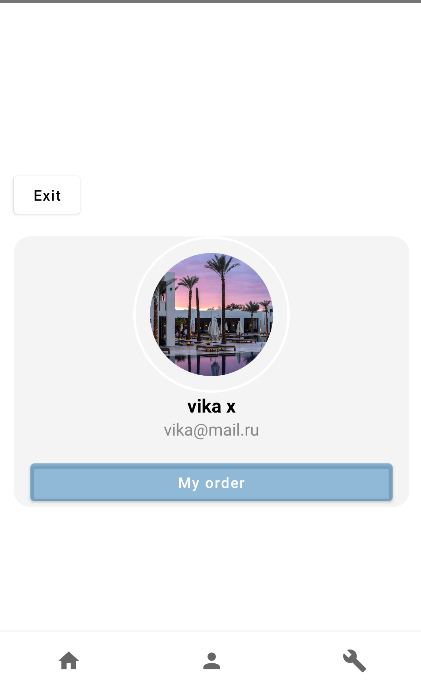
**Страница с оформлением заказа**

@@Composable  
fun BookingScreen(hotel: Hotel, navHostController: NavHostController) {  
 Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*background*(Color.White)  
 .*padding*(16.*dp*)  
 ,  
 verticalArrangement = Arrangement.Center,  
 horizontalAlignment = Alignment.CenterHorizontally  
 ) **{** Text(  
 text = "Booking",  
 fontSize = 24.*sp*,  
 fontWeight = FontWeight.Bold,  
 modifier = Modifier  
 .*padding*(16.*dp*)  
 )  
  
 TextField(  
 value =,  
 onValueChange = **{ }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*padding*(16.*dp*, 0.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*)),  
 singleLine = true,  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 placeholder = **{** Text(  
 text = "Rooms",  
 style = TextStyle(fontSize = 12.*sp*)  
 )  
 **}** )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
// Fetching the Local Context  
 val mContext = *LocalContext*.current  
  
 // Declaring integer values  
 // for year, month and day  
 val mYearFrom: Int  
 val mMonthFrom: Int  
 val mDayFrom: Int  
  
 // Initializing a Calendar  
 val mCalendarFrom = Calendar.getInstance()  
  
 // Fetching current year, month and day  
 mYearFrom = mCalendarFrom.get(Calendar.*YEAR*)  
 mMonthFrom = mCalendarFrom.get(Calendar.*MONTH*)  
 mDayFrom = mCalendarFrom.get(Calendar.*DAY\_OF\_MONTH*)  
  
 mCalendarFrom.*time* = Date()  
  
 // Declaring a string value to  
 // store date in string format  
 val mDateFrom = remember **{** *mutableStateOf*("") **}** // Declaring DatePickerDialog and setting  
 // initial values as current values (present year, month and day)  
 val mDatePickerDialogFrom = DatePickerDialog(  
 mContext,  
 **{** \_: DatePicker, mYear: Int, mMonth: Int, mDayOfMonth: Int **->** mDateFrom.value = "$mDayOfMonth/${mMonth+1}/$mYear"  
 **}**, mYearFrom, mMonthFrom, mDayFrom  
 )  
  
  
 // Creating a button that on  
 // click displays/shows the DatePickerDialog  
 Button(  
 colors = ButtonDefaults.buttonColors(  
 backgroundColor = (colorResource(id = R.color.*figma\_blue*)),  
 contentColor = Color.White  
 ),  
 onClick = **{** mDatePickerDialogFrom.show()  
  
 **}**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(16.*dp*, 16.*dp*, 16.*dp*, 0.*dp*)  
 .*height*(50.*dp*)  
 ) **{** Text("Open Date Picker")  
 **}** // Displaying the mDate value in the Text  
 Text(text = "Selected Date From: ${mDateFrom.value}", fontSize = 15.*sp*)  
  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
  
// Fetching the Local Context  
  
 // Declaring integer values  
 // for year, month and day  
 val mYear: Int  
 val mMonth: Int  
 val mDay: Int  
  
 // Initializing a Calendar  
 val mCalendar = Calendar.getInstance()  
  
 // Fetching current year, month and day  
 mYear = mCalendar.get(Calendar.*YEAR*)  
 mMonth = mCalendar.get(Calendar.*MONTH*)  
 mDay = mCalendar.get(Calendar.*DAY\_OF\_MONTH*)  
  
 mCalendar.*time* = Date()  
  
 // Declaring a string value to  
 // store date in string format  
 val mDate = remember **{** *mutableStateOf*("") **}** // Declaring DatePickerDialog and setting  
 // initial values as current values (present year, month and day)  
 val mDatePickerDialog = DatePickerDialog(  
 mContext,  
 **{** \_: DatePicker, mYear: Int, mMonth: Int, mDayOfMonth: Int **->** mDate.value = "$mDayOfMonth/${mMonth+1}/$mYear"  
 **}**, mYear, mMonth, mDay  
 )  
  
  
 // Creating a button that on  
 // click displays/shows the DatePickerDialog  
  
 Button(  
 colors = ButtonDefaults.buttonColors(  
 backgroundColor = (colorResource(id = R.color.*figma\_blue*)),  
 contentColor = Color.White  
 ),  
 onClick = **{** mDatePickerDialog.show()  
  
 **}**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(16.*dp*, 16.*dp*, 16.*dp*, 0.*dp*)  
 .*height*(50.*dp*)  
 ) **{** Text("Open Date Picker")  
 **}** // Displaying the mDate value in the Text  
 Text(text = "Selected Date To: ${mDate.value}", fontSize = 15.*sp*,)  
  
  
 Button(  
 colors = ButtonDefaults.buttonColors(  
 backgroundColor = (colorResource(id = R.color.*figma\_blue*)),  
 contentColor = Color.White  
 ),  
 onClick = **{**   
 }  
 **}**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(16.*dp*, 16.*dp*, 16.*dp*, 0.*dp*)  
 .*height*(50.*dp*)  
 ) **{** Text("Book")  
 **}  
 }** }



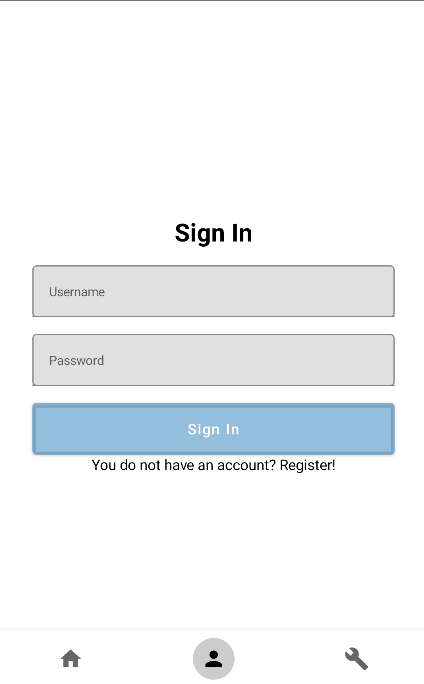
**Страница профиля**

@Composable  
fun Person() {  
 Column(  
 modifier = Modifier  
 .*background*(Color.White)  
 .*fillMaxSize*()  
 .*padding*(16.*dp*),  
 verticalArrangement = Arrangement.Top,  
 horizontalAlignment = Alignment.CenterHorizontally  
 ) **{** Image(  
 contentScale = ContentScale.FillBounds,  
 painter = painterResource(id = R.drawable.*img\_3*),  
 contentDescription = null,  
 modifier = Modifier  
 .*size*(120.*dp*)  
 .*clip*(*CircleShape*)  
 .*border*(2.*dp*, Color.Gray, *CircleShape*)  
 .*background*(Color.Gray)  
 .*padding*(8.*dp*)  
 )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 Text(  
 text = "vika@mail.ru",  
 fontSize = 18.*sp*,  
 fontWeight = FontWeight.Bold  
 )  
  
 Text(  
 text = "shailushai@example.com",  
 fontSize = 16.*sp*,  
 color = Color.Gray  
 )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 **}**}



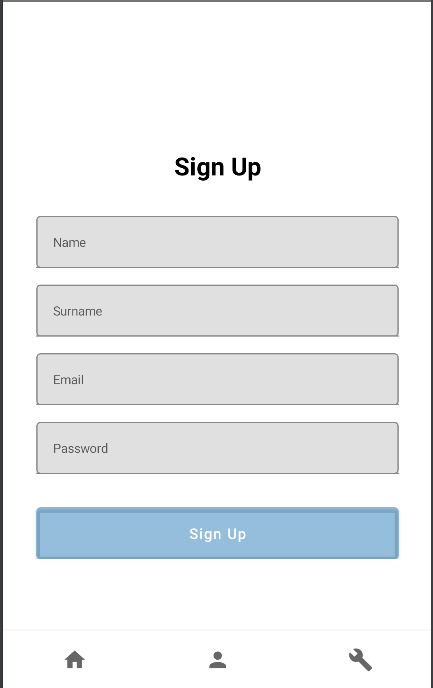
**Страница входа**

@Composable  
fun LoginScreen() {  
 var username by remember **{** *mutableStateOf*("") **}** var password by remember **{** *mutableStateOf*("") **}** Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*clip*(*RoundedCornerShape*(25.*dp*))  
 .*background*(colorResource(id = R.color.*figma*))  
 .*padding*(15.*dp*, 0.*dp*)  
 ,  
 verticalArrangement = Arrangement.Center,  
 horizontalAlignment = Alignment.CenterHorizontally  
 ) **{** Text1(text = "Sign In", fontSize = 24.*sp*, fontWeight = FontWeight.Bold)  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 BasicTextField(  
 value = username,  
 onValueChange = **{** username = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*padding*(8.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*))  
 .*padding*(8.*dp*),  
 singleLine = true,  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 )  
  
 BasicTextField(  
 value = password,  
 onValueChange = **{** password = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*padding*(8.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*))  
 .*padding*(8.*dp*),  
 singleLine = true,  
 visualTransformation = PasswordVisualTransformation(),  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 )**{  
  
 }** Button(  
 onClick = **{  
  
 }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(8.*dp*, 0.*dp*, 8.*dp*, 0.*dp*)  
 .*clip*(*RoundedCornerShape*(20.*dp*))  
 ) **{** Text1("Sign In")  
 **}  
 }**}



**Страница регистрации**

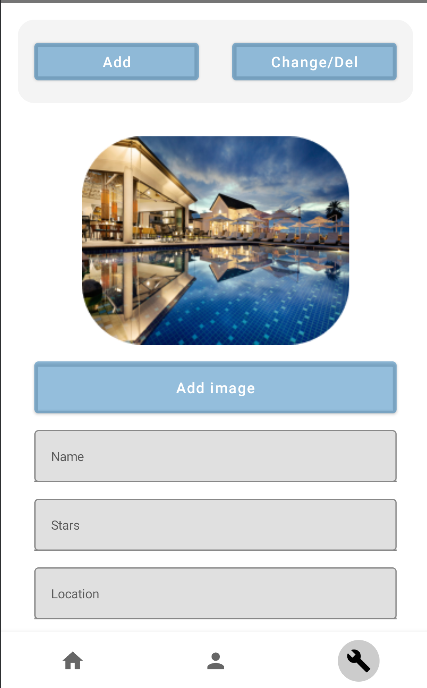
@Composable  
fun SignUpScreen() {  
 var username by remember **{** *mutableStateOf*("") **}** var password by remember **{** *mutableStateOf*("") **}** var sex by remember **{** *mutableStateOf*("") **}** var name by remember **{** *mutableStateOf*("") **}** var surname by remember **{** *mutableStateOf*("") **}** Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*clip*(*RoundedCornerShape*(25.*dp*))  
 .*background*(colorResource(id = R.color.*figma*))  
 .*padding*(15.*dp*, 0.*dp*)  
 ,  
 verticalArrangement = Arrangement.Center,  
 horizontalAlignment = Alignment.CenterHorizontally  
 ) **{** Text(text = "Sign Up", fontSize = 24.*sp*, fontWeight = FontWeight.Bold)  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 TextField(  
 value = username,  
 onValueChange = **{** username = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*))  
 .*padding*(0.*dp*),  
 singleLine = true,  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 placeholder = **{** Text(  
 text = "Username",  
 style = TextStyle(fontSize = 12.*sp*)  
 )  
 **}** )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 TextField(  
 value = name,  
 onValueChange = **{** name = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*)),  
 singleLine = true,  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 placeholder = **{** Text(  
 text = "Name",  
 style = TextStyle(fontSize = 12.*sp*)  
 )  
 **}** )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 TextField(  
 value = surname,  
 onValueChange = **{** surname = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*))  
 .*padding*(0.*dp*),  
 singleLine = true,  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 placeholder = **{** Text(  
 text = "Surname",  
 style = TextStyle(fontSize = 12.*sp*)  
 )  
 **}** )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 TextField(  
 value = sex,  
 onValueChange = **{** sex = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*))  
 .*padding*(0.*dp*),  
 singleLine = true,  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 placeholder = **{** Text(  
 text = "Sex",  
 style = TextStyle(fontSize = 12.*sp*)  
 )  
 **}** )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 TextField(  
 value = password,  
 onValueChange = **{** password = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*))  
 .*padding*(0.*dp*),  
 singleLine = true,  
 visualTransformation = PasswordVisualTransformation(),  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 placeholder = **{** Text(  
 text = "Password",  
 style = TextStyle(fontSize = 12.*sp*)  
 )  
 **}** )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 Button(  
 onClick = **{  
  
 }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*clip*(*RoundedCornerShape*(20.*dp*))  
 ) **{** Text("Sign Up")  
 **}  
 }**}



**Страница админа**

**Экран добавления товара**

@Composable  
@Preview  
fun AddPanel(){  
 var brand by remember **{** *mutableStateOf*("") **}** var model by remember **{** *mutableStateOf*("") **}** var description by remember **{** *mutableStateOf*("") **}** var price by remember **{** *mutableStateOf*("") **}** Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*background*(Color.White)  
 .*padding*(16.*dp*)  
 )**{** Box(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(150.*dp*)  
 .*background*(Color.Gray)  
 ) **{  
 }** Button(  
 onClick = **{  
 }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*clip*(*RoundedCornerShape*(20.*dp*))  
 .*padding*(16.*dp*)  
 ) **{** Text("Add image")  
 **}** TextField(  
 value = brand,  
 onValueChange = **{** brand = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*))  
 .*padding*(0.*dp*),  
 singleLine = true,  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 placeholder = **{** Text(  
 text = "Brand",  
 style = TextStyle(fontSize = 12.*sp*)  
 )  
 **}** )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 TextField(  
 value = model,  
 onValueChange = **{** model = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*))  
 .*padding*(0.*dp*),  
 singleLine = true,  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 placeholder = **{** Text(  
 text = "Model",  
 style = TextStyle(fontSize = 12.*sp*)  
 )  
 **}** )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 TextField(  
 value = description,  
 onValueChange = **{** description = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(100.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*))  
 .*padding*(0.*dp*),  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 placeholder = **{** Text(  
 text = "Description",  
 style = TextStyle(fontSize = 12.*sp*)  
 )  
 **}** )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 TextField(  
 value = price,  
 onValueChange = **{** price = **it }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(50.*dp*)  
 .*border*(1.*dp*, Color.Gray, *RoundedCornerShape*(4.*dp*))  
 .*padding*(0.*dp*),  
 singleLine = true,  
 keyboardOptions = KeyboardOptions(  
 keyboardType = KeyboardType.Text,  
 imeAction = ImeAction.Next  
 ),  
 keyboardActions = KeyboardActions(  
 onNext = **{  
  
 }** ),  
 placeholder = **{** Text(  
 text = "Price",  
 style = TextStyle(fontSize = 12.*sp*)  
 )  
 **}** )  
  
 Button(  
 onClick = **{  
  
 }**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*clip*(*RoundedCornerShape*(20.*dp*))  
 .*padding*(16.*dp*)  
 ) **{** Text("Add sneaker")  
 **}  
 }**}



**Страница с редактирование и удалением товара**

@Composable  
fun ChangePanel(navHostController: NavHostController) {  
 Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*background*(Color.White)  
 )**{** Row **{** LazyColumn(  
 modifier = Modifier  
 .*fillMaxSize*()  
 ) **{** *itemsIndexed*(  
 *listOf*(  
 SneakerItem(R.drawable.*sneaker*, "Jordan", 159.99),  
 SneakerItem(R.drawable.*sneaker*, "Jordan", 159.99),  
 SneakerItem(R.drawable.*trash*, "Nike", 179.99),  
 )  
 )**{**\_, item**->** CardSneakerForChange(item = item, navHostController)  
  
 **}  
 }  
 }  
 }**}

