**ТЕСТОВЫЙ НАБОР, ТЕСТ-КЕЙСЫ, UI-ТЕСТЫ, UNIT-ТЕСТЫ**

1. **Тестовые наборы**
   1. **UI-тесты**
2. Экран Authorization

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Отображение экрана Authorization | Приложение отобразило экран Authorization |
| Отображение сообщения о пустых полях ввода на экране | Приложение отобразило сообщение, что поля ввода пустые |
| Отображение сообщения о некорректных данных для авторизации | Приложение отобразило сообщение, что телефон или пароль неверный |

1. Экран VisitingWorkers

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Отображение экрана VisitingWorkers после успешной авторизации главного по смене на экране Authorization | Приложение отобразило экран VisitingWorkers |

1. Экран TasksWorker

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Отображение экрана TasksWorker после успешной авторизации работника на экране Authorization | Приложение отобразило экран TasksWorker |

**2.1. Unit-тесты**

1. AuthorizationViewModel

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Работоспособность метода onPhoneChange | Метод onPhoneChange положило значение в свойство VM |
| Работоспособность метода onPasswordChange | Метод onPasswordChange положило значение в свойство VM |
| Инициализация свойства navigateTo | Свойство navigateTo инициализировалось как null |

2. TasksWorkerViewModel

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Инициализация свойства taskList | Свойство taskList инициализировалось пустым списком |

3. MessagesViewModel

|  |  |
| --- | --- |
| Тест-кейсы | |
| Название | Ожидаемый результат |
| Инициализация свойства isError | Свойство isError инициализировалось как false |

1. Тест-кейсы

* Название: Отображение экрана Authorization

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleInstrumentedTest»

3. Напротив функции Authorization\_IsDisplayed нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что экран Authorization отобразился

* Название: Отображение сообщения о пустых полях ввода на экране

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleInstrumentedTest»

3. Напротив функции Authorization\_Toast\_EmptyTextFolder\_IsDisplayed нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что на экран Authorization отобразился сообщение о пустых полях

* Название: Отображение сообщения о некорректных данных для

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleInstrumentedTest»

3. Напротив функции Authorization\_Toast\_IncorrectSignIn\_IsDisplayed нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что на экране Authorization отобразилось сообщение о том, что телефон или пароль неверный

* Название: Отображение экрана VisitingWorkers после успешной авторизации главного по смене на экране

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleInstrumentedTest»

3. Напротив функции Authorization\_SignIn\_To\_VisitingWorkers нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что экран VisitingWorkers отобразился после успешной авторизации на экране Authorization

* Название: Отображение экрана TasksWorker после успешной авторизации работника на экране Authorization

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleInstrumentedTest»

3. Напротив функции Authorization\_SignIn\_To\_TaskWorker нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что экран TasksWorker отобразился после успешной авторизации на экране Authorization

* Название: Работоспособность метода onPhoneChange

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleUnitTest »

3. Напротив функции TestMethod\_AuthorizationViewModel\_onPhoneChange нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что значение поля в VM совпало с ожидаемым значением

* Название: Работоспособность метода onPasswordChange

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleUnitTest »

3. Напротив функции TestMethod\_AuthorizationViewModel\_onPasswordChange нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что значение поля в VM совпало с ожидаемым значением

* Название: Инициализация свойства navigateTo

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleUnitTest »

3. Напротив функции TestProperty\_AuthorizationViewModel\_NavigateToValue нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что в свойстве navigateTo лежит null

* Название: Инициализация свойства taskList

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleUnitTest »

3. Напротив функции TestProperty\_TasksWorkerViewModel\_TaskListValuee нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что в свойстве taskList лежит пустой список

* Название: Инициализация свойства isError

Шаги:

1. Открыть «Android Studio»

2. Открыть файл «ExampleUnitTest »

3. Напротив функции TestProperty\_MessagesViewModel\_isErrorValue нажать зеленый треугольник (кнопка запуска теста)

Ожидаемый результат: функция вернет то, что в свойстве isError лежит false

1. UI-тесты

1. Экран Authorization

| № | Название теста | Описание | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Отображение экрана Authorization | Проверяет, что приложение отображает экран Authorization | Приложение отобразило экран Authorization. |
| @Test fun Authorization\_IsDisplayed() {  composeTestRule.setContent {  val auth: Auth = client.auth  val postgrest: Postgrest = client.postgrest  val workerRep: WorkerRepository = WorkerRepositoryImpl(postgrest)  val authRep: AuthenticationRepository = AuthenticationRepositoryImpl(auth)  authVM = AuthorizationViewModel(authRep, workerRep)  TestNavigation(AuthorizationDestination.route)  }  composeTestRule.onNodeWithText("Авторизация").assertIsDisplayed() } | | | |
| 2 | Отображение сообщения о пустых полях ввода на экране | Проверяет, что приложение отображает на экране Authorization сообщение о пустых текстовых полях | Приложение отобразило сообщение, что поля ввода пустые |
| @Test fun Authorization\_Toast\_EmptyTextFolder\_IsDisplayed() {  composeTestRule.setContent {  val auth: Auth = client.auth  val postgrest: Postgrest = client.postgrest  val workerRep: WorkerRepository = WorkerRepositoryImpl(postgrest)  val authRep: AuthenticationRepository = AuthenticationRepositoryImpl(auth)  authVM = AuthorizationViewModel(authRep, workerRep)  TestNavigation(AuthorizationDestination.route)  }  composeTestRule.onNodeWithText("Вход").performClick()  composeTestRule.onNodeWithText("Поля не должны быть пустыми").isDisplayed() } | | | |
| 3 | Отображение сообщения о некорректных данных для авторизации | Проверяет, что приложение отображает на экране Authorization сообщение о неверном телефоне или пароле | Приложение отобразило сообщение, что телефон или пароль неверный. |
| @Test fun Authorization\_Toast\_IncorrectSignIn\_IsDisplayed() {  composeTestRule.setContent {  val auth: Auth = client.auth  val postgrest: Postgrest = client.postgrest  val workerRep: WorkerRepository = WorkerRepositoryImpl(postgrest)  val authRep: AuthenticationRepository = AuthenticationRepositoryImpl(auth)  authVM = AuthorizationViewModel(authRep, workerRep)  authVM.phoneValue = "+3219"  authVM.passwordValue = "123460"  TestNavigation(AuthorizationDestination.route)  }  composeTestRule.onNodeWithText("Вход").performClick()  composeTestRule.onNodeWithText("Неверный телефон или пароль").isDisplayed() } | | | |
| Результаты тестов | | | |
|  | | | |

|  |
| --- |
|  |

2. Экран VisitingWorkers

| № | Название теста | Описание | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Отображение экрана VisitingWorkers после успешной авторизации главного по смене на экране Authorization | Проверяет, что приложение отображает экран VisitingWorkers после успешной авторизации на экране Authorization | Приложение отобразило экран VisitingWorkers |
| @Test fun Authorization\_SignIn\_To\_VisitingWorkers() {  composeTestRule.setContent {  val auth: Auth = client.auth  val postgrest: Postgrest = client.postgrest  val workerRep: WorkerRepository = WorkerRepositoryImpl(postgrest)  val authRep: AuthenticationRepository = AuthenticationRepositoryImpl(auth)  authVM = AuthorizationViewModel(authRep, workerRep)  authVM.phoneValue = "+79290509319"  authVM.passwordValue = "123460"  TestNavigation(AuthorizationDestination.route)   }  composeTestRule.onNodeWithText("Вход").performClick()  composeTestRule.onNodeWithText("К задачам").isDisplayed() } | | | |

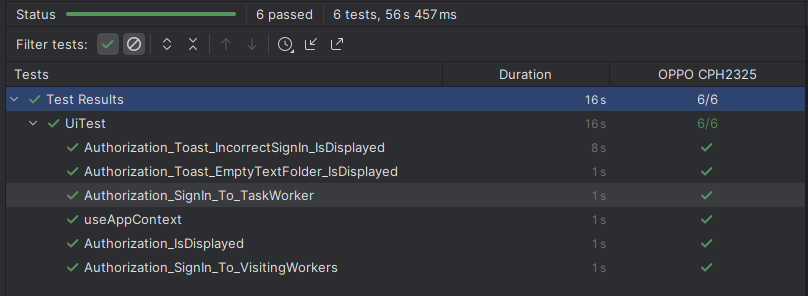
|  |
| --- |
| Результаты тестов |
|  |

3. Экран TasksWorker

| № | Название теста | Описание | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Отображение экрана TasksWorker после успешной авторизации главного по смене на экране Authorization | Проверяет, что приложение отображает экран TasksWorker после успешной авторизации на экране Authorization | Приложение отобразило экран TasksWorker |
| @Test fun Authorization\_SignIn\_To\_TaskWorker() {  composeTestRule.setContent {  val auth: Auth = client.auth  val postgrest: Postgrest = client.postgrest  val workerRep: WorkerRepository = WorkerRepositoryImpl(postgrest)  val authRep: AuthenticationRepository = AuthenticationRepositoryImpl(auth)  authVM = AuthorizationViewModel(authRep, workerRep)  authVM.phoneValue = "+79290509325"  authVM.passwordValue = "123466"  TestNavigation(AuthorizationDestination.route)   }  composeTestRule.onNodeWithText("Вход").performClick()  composeTestRule.onNodeWithText("Сообщения").isDisplayed() } | | | |

|  |
| --- |
| Результаты тестов |
|  |

Прохождение всех тестов:



1. UNIT-тесты

1. AuthorizationViewModel

| № | Название теста | Описание | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Работоспособность метода onPhoneChange | Проверяет, что приложение записало значение телефона через метод в AuthorizationViewModel | Метод onPhoneChange положило значение в свойство VM |
| @Test fun TestMethod\_AuthorizationViewModel\_onPhoneChange() {  val mockWorkerRepository =  mockk<WorkerRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  val mockAuthRepository =  mockk<AuthenticationRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  authVM = AuthorizationViewModel(mockAuthRepository, mockWorkerRepository)  authVM.onPhoneChange("+79290509325")  assertEquals("+79290509325", authVM.phoneValue) } | | | |
| 2 | Работоспособность метода onPasswordChange | Проверяет, что приложение записало значение пароля через метод в AuthorizationViewModel | Метод onPasswordChange положило значение в свойство VM |
| @Test fun TestMethod\_AuthorizationViewModel\_onPasswordChange() {  val mockWorkerRepository =  mockk<WorkerRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  val mockAuthRepository =  mockk<AuthenticationRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  authVM = AuthorizationViewModel(mockAuthRepository, mockWorkerRepository)  authVM.onPasswordChange("123466")  assertEquals("123466", authVM.passwordValue) } | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Ожидаемый результат |
| 3 | Инициализация свойства navigateTo | Проверяет, что приложение инициализировало значение свойства navigateTo в AuthorizationViewModel как null | Свойство navigateTo инициализировалось как null |
| @Test fun TestProperty\_AuthorizationViewModel\_NavigateToValue() {  val mockWorkerRepository =  mockk<WorkerRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  val mockAuthRepository =  mockk<AuthenticationRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  authVM = AuthorizationViewModel(mockAuthRepository, mockWorkerRepository)  assertEquals(null, authVM.navigateToValue) } | | | |
| Результаты тестов | | | |
|  | | | |

2. TasksWorkerViewModel

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Ожидаемый результат |
| 1 | Инициализация свойства taskList | Проверяет, что приложение инициализировало значение свойства taskList в TasksWorkerViewModel пустым списком | Свойство taskList инициализировалось пустым списком |
| @Test fun TestProperty\_TasksWorkerViewModel\_TaskListValuee() {  val mockTaskRepository =  mockk<TaskRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  taskVM = TasksWorkerViewModel(mockTaskRepository)  assertEquals(listOf<Task>(), taskVM.taskListValue)  } | | | |
| Результаты тестов | | | |
|  | | | |

3. MessagesViewModel

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Ожидаемый результат |
| 1 | Инициализация свойства isError | Проверяет, что приложение инициализировало значение свойства isError в MessagesViewModel как false | Свойство isError инициализировалось как false |
| @Test fun TestProperty\_MessagesViewModel\_isErrorValue() {  val mockMessageRepository =  mockk<MessagesRepository>(relaxed = true) // Также заменяем WorkerRepository на mock  messageVM = MessagesViewModel(mockMessageRepository)  assertEquals(false, messageVM.isErrorValue)  } | | | |
| Результаты тестов | | | |
|  | | | |

Прохождение всех тестов:

