

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЕВМ

ОТЧЕТ
по «UI-тестирование» практике
Тема: Gmail.com

Студент гр. 3388

Тимошук Е. Потапов Р.
Кулач Д.

Руководитель

Шевелева А.М.

Санкт-Петербург

2025

ЗАДАНИЕ
НА «UI-тестирование» ПРАКТИКУ

Студенты Тимошук Е. Потапов Р. Кулач Д.

Группа 3388

Тема практики: **Gmail.com**

Задание на практику:

Написать «UI-тестирование» для системы Gmail

Сроки прохождения практики: 26.06.2025 – 05.07.2025

Дата сдачи отчета: 05.07.2025

Дата защиты отчета: 05.07.2025

Студент

Тимошук Е. Потапов Р.
Кулач Д.

Руководитель

Шевелева А.М.

АННОТАЦИЯ

В данном отчете представлены результаты прохождения практики, посвященной UI-тестированию интерфейса Gmail. Были рассмотрены основные элементы пользовательского интерфейса, проведено функциональное и юзабилити-тестирование, а также проанализирована работа сервиса в разных браузерах и на различных устройствах.

В ходе тестирования использовались ручные методы проверки и инструменты разработчика (DevTools). Составлены тест-кейсы, выявлены возможные баги и предложены рекомендации по улучшению интерфейса.

Отчет содержит описание методологии тестирования, примеры тест-кейсов с результатами, список обнаруженных проблем и выводы по проделанной работе

SUMMARY

This report presents the results of the practice dedicated to UI testing of the Gmail interface. The main elements of the user interface were reviewed, functional and usability testing was conducted, and the service's performance in different browsers and on various devices was analyzed.

During the testing, manual verification methods and developer tools (DevTools) were used. Test cases were created, possible bugs were identified, and recommendations for improving the interface were proposed.

The report contains a description of the testing methodology, examples of test cases with results, a list of detected issues, and conclusions on the work performed.

Содержание

Цель работа	5
Выполнение работы	6
ЧЕК ЛИСТ	6
ТЕСТ (УСТАНОВКА ФИЛЬТРОВ ДЛЯ ПИСЕМ)	8
ТЕСТ (ИЗМЕНЕНИЕ ФИО ПОЛЬЗОВАТЕЛЯ)	10
ТЕСТ (УДАЛЕНИЕ ПИСЬМА)	12
ТЕСТ (ДОБАВЛЕНИЕ КОНТАКТА)	14
ТЕСТ (СМЕНА ЯЗЫКА)	16
ТЕСТ (ОТПРАВКА ПИСЬМА НЕСУЩ. АДРЕССУ)	18
ТЕСТ (ПОСТАНОВЛЕНИЕ ЗАДАЧИ)	20
ТЕСТ (ПОМЕТИТЬ ПИСЬМО КАК НЕПРОЧИТАННОЕ)	22
ТЕСТ (ПОИСК, ПО КЛЮЧЕВЫМ СЛОВАМ)	24
ТЕСТ (ПРОВКА ВОССТАНОВЛЕНИЯ СЕССИИ ПОСЛЕ РАЗРЫВА СОЕДИНЕНИЯ)	26
UML – ДИАГРАММА	28
Вывод	30
Список используемой литературы:	31

Цель работа

Изучение методов UI-тестирования на примере Gmail.

Задачи:

- Анализ интерфейса Gmail.
- Проверка юзабилити и функциональности.
- Составление тест-кейсов.

Выполнение работы

ЧЕК ЛИСТ

Тестирование системы Mail (<https://mail.ru/>)

Тесты	Шаги теста	Ожидаемый результат
1. Установка фильтров для писем	1. Авторизоваться в почтовой системе. 2. Перейти в раздел «Настройки» → «Фильтры и правила» . 3. Нажать «Создать новый фильтр» . 4. Задать условия фильтрации: Поле: "Отправитель" → ввести example@domain.com. Действие: "Переместить в папку «Важное»". 5. Сохранить фильтр. 6. Отправить тестовое письмо с адреса example@domain.com. 7. Проверить папку «Важное» — письмо должно появиться там.	Письмо от example@domain.com автоматически перемещается в папку «Важное».
2. Изменение имени и фамилии пользователя	1. Авторизоваться в системе. 2. Перейти в «Настройки» → «Профиль» . 3. В полях «Имя» и «Фамилия» ввести новые значения (например, "Иван Иванов"). 4. Нажать «Сохранить» . 5. Открыть отправленное письмо	Письмо исчезает из исходной папки и появляется в «Корзине». При восстановлении — возвращается на прежнее место.
3. Удаление письма	1. Авторизоваться в системе. 2. Выбрать письмо во «Входящих» 3. Нажать кнопку «Удалить» . 4. Перейти в раздел «Корзина» — убедиться, что письмо там. Вариант восстановления: 1. Выбрать письмо в «Корзине» → нажать «Восстановить» . 2. Проверить исходную папку (например, «Входящие»).	Сообщение удалено или успешно восстановлено
1. Добавление контакта	1. Авторизоваться в почтовой системе. 2. Перейти в раздел «Контакты» . 3. Нажать кнопку «Новый контакт» . 4. Заполнить поля: Имя: Даниил Email: kulachdv@gmail.com 5. Нажать «Сохранить» .	В списке контактов появился «Даниил»
2. Проверка	1. Авторизоваться в системе	Система сохраняет

восстановления сессии после разрыва соединения	2. Начать составление письма (ввести текст, добавить вложение) 3. Имитировать разрыв соединения 4. Подождать 1-2 минуты 5. Восстановить соединение 6. Проверить: Сохранение черновика Возможность продолжить работу Состояние интерфейса	все введенные данные
3. Смена языка	1. Авторизоваться в системе 2. нажать на “Управление аккаунтом Google” 3. Нажать на “Личная информация” 4. найти “Общие настройки веб-интерфейса” 5. Сменить язык	Язык системы был изменен
4. Отправка письма несуществующему адресату (проверка ошибки).	1. Авторизоваться в системе 2. Нажмите «Написать письмо» . 3. В поле «Кому» введите: user@ 4. Попробуйте отправить письмо.	Система дает возможность отправить письмо любому пользователю (даже несуществ.)
1. Постановление задачи	1. Авторизоваться в системе 2. Нажать на “Задачи” 3. Нажать на “Добавить задачу” 4. Придумать название 5. Сохранить	Система создает задачу
2. Пометить письмо как прочитанное/не прочитанное	1. Авторизоваться в системе 2. Найдите непрочитанное письмо в папке «Входящие». 3. Кликните на письмо Правой кнопкой мыши (ПКМ) → «Пометить как прочитанное».	Статус письма меняется на «Прочитано»
3. Поиск, по ключевым словам, в письмах	1. Авторизоваться в системе. 2. В глобальное поле поиска поочередно ввести: - «Здравствуйте» — существующее слово; - «Несуществующее Слово» — вымышленное слово. После каждого ввода нажать Enter. 3. Установить фильтр «Тема» и повторить шаг 2. 4. Переключить фильтр на «Слова в письме» и снова повторить шаг 2.	«Здравствуйте» - > показываются только письма, где слово найдено в выбранной области; результатов > 0. «Несуществующее Слово» -> «Нет результатов», список пуст.

ТЕСТ (УСТАНОВКА ФИЛЬТРОВ ДЛЯ ПИСЕМ)

Тест автоматизирует сценарий создания фильтра в Gmail, который гарантирует, что письма от указанного отправителя никогда не попадают в спам.

Цель теста

Проверить функциональность создания фильтра:

1. Для заданного email-адреса (`no-reply@etu.ru`)
2. С опцией "Никогда не отправлять в спам"
3. Через стандартный интерфейс настроек Gmail

Пошаговая логика работы

Шаг 1: Авторизация в Gmail

- ```
InboxPage inbox = auth(USER_EMAIL, USER_PASSWORD);
```
- Вызов метода `auth()` из базового класса `BaseTest`
  - Возвращает объект страницы входящих писем (`InboxPage`)

Шаг 2: Переход к настройкам фильтров

- ```
SettingsPage settings = inbox.clickSettingsIcon();  
settings.clickAllSettings();  
FiltersPage filters = settings.openFiltersPage();
```
1. Клик по иконке настроек (шестерёнка)
 2. Переход в "Все настройки"
 3. Открытие раздела "Фильтры и заблокированные адреса"

Шаг 3: Создание фильтра

- ```
filters.clickCreateNewFilter();
filters.enterFromAddress(FROM_ADDRESS);
filters.clickCreateFilterLink();
filters.clickNeverSpamOption();
filters.clickFinalCreateFilter();
```
1. Начало создания нового фильтра
  2. Ввод email отправителя в поле "От"
  3. Переход к настройке действий фильтра



4. Активация опции "Никогда не отправлять в спам"
5. Сохранение фильтра

## Настройки

Общие Ярлыки Папка "Входящие" Аккаунты и импорт [Фильтры и заблокированные адреса](#)

Следующие фильтры применяются ко всем входящим письмам:

☐

Соответствует: **from:(pelmeshka@gmail.com)**

Действия: Никогда не отправлять в спам

Выбрать: [Все](#), [Ни одного](#)

Экспорт

Удалить

Рисунок 1 - Создание фильтра

## ТЕСТ (ИЗМЕНЕНИЕ ФИО ПОЛЬЗОВАТЕЛЯ)

Объяснение работы теста `ProfileNameTest`

Тест проверяет сценарий изменения имени и фамилии в профиле Google через интерфейс Gmail. Разберём пошагово:

### 1. Инициализация констант

```
private static final String USER_EMAIL = "логин@gmail.com";
private static final String USER_PASSWORD = "пароль";
private static final String NEW_FIRST = "Родион";
private static final String NEW_LAST = "Потапов";
```

- Учётные данные для входа в Gmail
- Новые значения имени и фамилии для теста

### 2. Структура теста `shouldUpdateProfileName()`

#### Шаг 1: Аутентификация

```
InboxPage inbox = auth(USER_EMAIL, USER_PASSWORD);
```

- Вызов метода `auth()` из `BaseTest` (предположительно реализует вход в Gmail)
- Возвращает объект страницы входящих писем (`InboxPage`)

#### Шаг 2: Переход в настройки профиля

```
ProfileMenuPage menu = inbox.openProfileMenu();
String originalHandle = driver.getWindowHandle();
MyAccountPage myAcc = menu.goToMyAccount();
```

1. `inbox.openProfileMenu()`: Открывает выпадающее меню профиля (иконка в правом верхнем углу)
2. `driver.getWindowHandle()`: Запоминает идентификатор текущей вкладки (Gmail)
3. `menu.goToMyAccount()`:
  - Кликает пункт "Мой аккаунт"
  - Открывает новую вкладку с настройками Google-аккаунта
  - Возвращает объект страницы `MyAccountPage`

### Шаг 3: Изменение имени

```
myAcc.openPersonalInfo()
.openNameEdit()
.updateName(NEW_FIRST, NEW_LAST)
.returnToOriginal(originalHandle);
```

Цепочка действий:

1. ``openPersonalInfo()``: Открывает раздел "Личная информация"
2. ``openNameEdit()``: Запускает форму редактирования имени
3. ``updateName(NEW_FIRST, NEW_LAST)``:
  - Вводит новые имя и фамилию
  - Сохраняет изменения
4. ``returnToOriginal(originalHandle)``:
  - Закрывает текущую вкладку ("Мой аккаунт")
  - Возвращает фокус на исходную вкладку (Gmail)

← Имя

Имя будет изменено во всех сервисах, где используется аккаунт Google. Указанное ранее имя может по-прежнему быть доступно для поиска или показываться в старых сообщениях. [Подробнее...](#)


Имя

Евгений

Фамилия

Тимошук

**Кто может видеть ваше имя**

 Эту информацию смогут увидеть любые пользователи, которые будут общаться с вами или просматривать созданный вами контент в сервисах Google. [Подробнее...](#)

[Отмена](#) [Сохранить](#)

Рисунок 2 – Форма по заполнение ФИО

## ТЕСТ (УДАЛЕНИЕ ПИСЬМА)

Тест автоматизирует сценарий удаления первого непрочитанного письма из ящика Gmail с проверкой изменения количества непрочитанных сообщений.

Логика работы теста

### 1. Авторизация в системе:

```
InboxPage inbox = auth(USER_EMAIL, USER_PASSWORD);
```

- Выполняется вход в аккаунт Gmail с использованием учётных данных

- Возвращается объект страницы входящих писем (`InboxPage`)

### 2. Проверка непрочитанных писем:

```
int before = inbox.getUnreadCount();
```

```
if (before == 0) {
```

```
System.out.println("Нет непрочитанных писем для удаления.");
```

```
return;
```

```
}
```

- Получает текущее количество непрочитанных писем

- Если писем нет, тест завершается с сообщением (без ошибки)

### 3. Выбор и удаление письма:

```
inbox.selectFirstUnread(); // Выделение первого непрочитанного
```

```
inbox.clickDelete(); // Удаление выделенного письма
```

- Находит первое непрочитанное письмо в списке и выделяет его

- Нажимает кнопку удаления (иконка корзины)

### 4. Проверка результата:

```
inbox.waitForUnreadCountLessThan(before);
```

- Ожидает уменьшения количества непрочитанных писем

- Неявная проверка: если количество не изменится - тест упадёт по таймауту

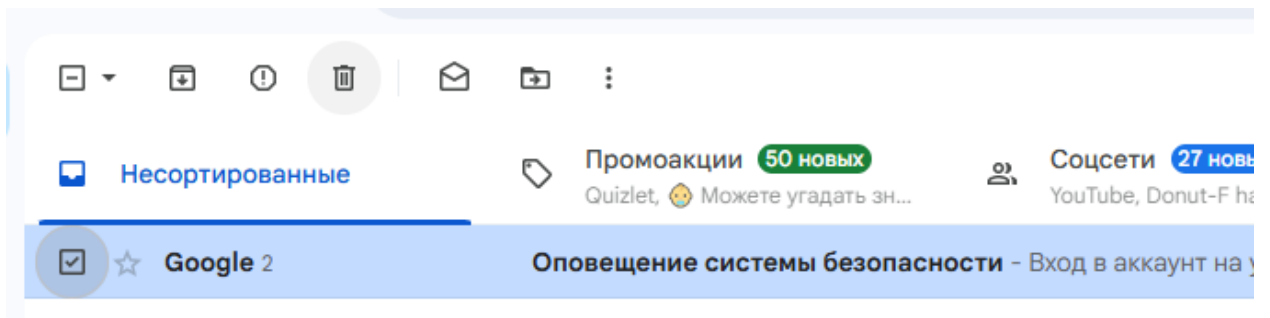


Рисунок 3 - Удаление письма

## ТЕСТ (ДОБАВЛЕНИЕ КОНТАКТА)

Тест автоматизирует сценарий создания нового контакта в Google Contacts через интерфейс Gmail. Разберём его логику по шагам:

### 1. Инициализация данных

```
private static final String USER_EMAIL = "логин@gmail.com";
private static final String USER_PASSWORD = "пароль";
private static final String CONTACT_NAME = "Родион";
private static final String CONTACT_EMAIL =
"rodionpotapov@gmail.com";
```

- Учётные данные для входа
- Имя и email создаваемого контакта

### 2. Шаги выполнения теста

#### Шаг 1: Авторизация в Gmail

```
InboxPage inbox = auth(USER_EMAIL, USER_PASSWORD);
```

- Стандартный вход через метод `auth()` из `BaseTest`
- Возвращает объект страницы входящих (`InboxPage`)

#### Шаг 2: Переход в раздел контактов\*\*

```
ContactsPage contacts = inbox.openContactsPage();
```

- Использует метод `openContactsPage()` для:
- Открытия меню приложений Google (квадрат из точек)
- Выбора иконки "Контакты"
- Перехода на новую вкладку с интерфейсом Google Contacts

#### Шаг 3: Создание нового контакта

```
contacts.clickNewContact();
contacts.enterName(CONTACT_NAME);
contacts.enterEmail(CONTACT_EMAIL);
contacts.saveContact();
```

1. `clickNewContact()`: Нажимает кнопку "Создать контакт"
2. `enterName()`: Вводит имя в поле "Имя"
3. `enterEmail()`: Добавляет email-адрес

#### 4. `saveContact()`: Сохраняет контакт

##### Шаг 4: Подтверждение сохранения

```
contacts.waitForSaveAndReturn();
```

- Ожидает визуального подтверждения успешного сохранения
- Закрывает форму редактирования (возврат к списку контактов)

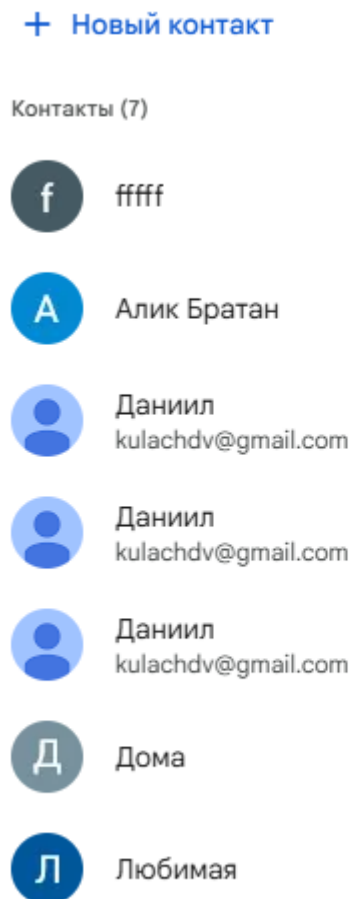


Рисунок 4 - Добавление Контакта

## ТЕСТ (СМЕНА ЯЗЫКА)

Цель теста

Проверить функциональность смены языка интерфейса:

1. Смена на указанный язык (`en-GB` - английский Великобритания)
2. Сохранение изменений через настройки
3. Проверка работы без UI-проверок (базовая версия)

Пошаговая логика работы

Шаг 1: Авторизация в Gmail

```
InboxPage inbox = auth(USER_EMAIL, USER_PASSWORD);
```

- Стандартный вход через метод `auth()` из `BaseTest`
- Возвращает объект страницы входящих писем

Шаг 2: Открытие полных настроек

```
SettingsPage settings = inbox.clickSettingsIcon();
```

```
settings.clickAllSettings();
```

1. Клик по иконке настроек (шестерёнка в правом верхнем углу)
2. Переход в раздел "Все настройки" (полная версия настроек)

Шаг 3: Изменение языка

```
settings.selectLanguage(LANGUAGE_VALUE);
```

```
settings.clickSaveChanges();
```

1. Выбор языка из выпадающего списка по коду `en-GB`
2. Сохранение изменений кнопкой "Сохранить"




## ← Language



Your preferred language for Google services and other languages that you might understand. [Learn more](#) ⓘ

Changes to your preferred language are reflected on the web. Google might use your language info to show you more relevant content on apps and services. To change the preferred language for mobile apps, go to the language settings on your device.

**Preferred language**

English  
Zimbabwe 

**Other languages**

Русский (Russian)  
 Added for you 

Save

+ Add another language

Рисунок 5 - Смена языка

## ТЕСТ (ОТПРАВКА ПИСЬМА НЕСУЩ. АДРЕССУ)

Тест автоматизирует сценарий отправки электронного письма через интерфейс Gmail. Разберём его работу детально:

Проверить полный цикл отправки письма:

1. Авторизация в системе
2. Создание нового письма
3. Заполнение обязательных полей
4. Отправка письма
5. Подтверждение отправки (неявное)

Пошаговая логика работы

Шаг 1: Авторизация

```
LoginPage login = new LoginPage(driver);
login.enterEmail(EMAIL);
login.clickEmailNext();
login.enterPassword(PASSWORD);
login.clickPasswordNext();
```

1. Создание объекта страницы авторизации
2. Ввод email
3. Переход к полю пароля
4. Ввод пароля
5. Завершение входа

Шаг 2: Инициация создания письма

```
InboxPage inbox = new InboxPage(driver);
inbox.clickCompose();
- Создание объекта страницы входящих писем
- Клик по кнопке "Написать" (Compose)
```

Шаг 3: Заполнение и отправка письма

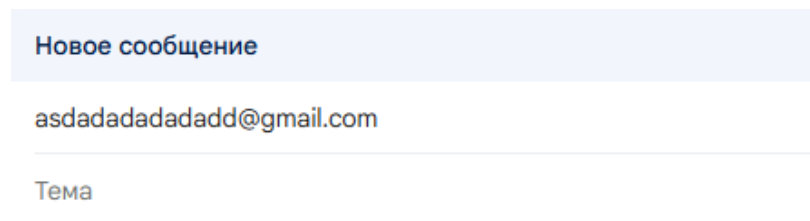
```
ComposePage compose = new ComposePage(driver);
compose.enterRecipient(ТО);
compose.enterSubject(SUBJECT);
```

```
compose.enterBody(BODY);
```

```
compose.clickSend();
```

1. Создание объекта страницы составления письма
2. Ввод адреса получателя
3. Ввод темы письма
4. Ввод текста письма
5. Клик по кнопке "Отправить"

**Тест является не совсем корректным, потому что письмо можно отправить любому выдуманному пользователю**



Новое сообщение

asdadadadadadd@gmail.com

Тема

Рисунок 6 - отправка письма

## ТЕСТ (ПОСТАНОВЛЕНИЕ ЗАДАЧИ)

Тест автоматизирует сценарий создания новой задачи в Google Tasks через интерфейс Gmail. Разберём его работу:

Цель теста

Проверить функционал добавления задач:

1. Авторизация в Gmail
2. Открытие интерфейса Google Tasks
3. Создание задачи с указанным названием
4. Базовое подтверждение операции

Пошаговая логика работы

Шаг 1: Авторизация в Gmail

```
InboxPage inbox = auth(USER_EMAIL, USER_PASSWORD);
```

- Использует метод `auth()` из базового класса `BaseTest`
- Возвращает объект страницы входящих писем

Шаг 2: Открытие интерфейса задач

```
TasksPage tasks = inbox.openTasksPage();
```

- Вызывает метод `openTasksPage()` который:
- Находит иконку Google Tasks (обычно в правом сайдбаре)
- Кликает по ней, открывая интерфейс задач
- Часто реализуется через переключение на iframe

Шаг 3: Создание задачи

```
tasks.clickAddTaskButton();
```

```
tasks.enterTaskTitle(TASK_TITLE);
```

```
tasks.submitTask();
```

1. Клик по кнопке "+" или "Добавить задачу"
2. Ввод названия задачи в текстовое поле
3. Подтверждение создания (обычно через Enter или кнопку "Сохранить")

Шаг 4: Возврат в основной интерфейс

```
tasks.switchToDefaultContent();
```

- Переключает контекст драйвера обратно на основную страницу
- Необходимо для корректной работы последующих тестов

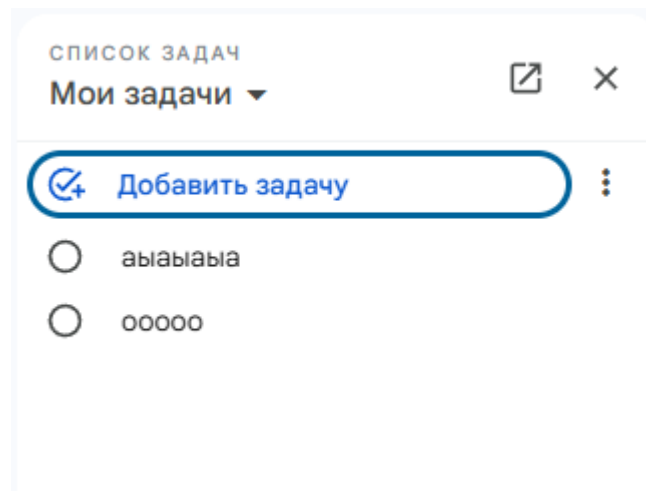


Рисунок 7 - Добавление задачи

## ТЕСТ (ПОМЕТИТЬ ПИСЬМО КАК НЕПРОЧИТАННОЕ)

Тест автоматизирует сценарий пометки первого непрочитанного письма как прочитанного в интерфейсе Gmail. Разберём его работу детально:

Цель теста

Проверить функционал изменения статуса письма:

1. Найти первое непрочитанное письмо
2. Изменить его статус на "прочитанное"
3. Снять выделение с письма
4. Убедиться в отсутствии ошибок выполнения

Пошаговая логика работы

Шаг 1: Авторизация в Gmail

```
LoginPage login = new LoginPage(driver);
login.enterEmail(EMAIL);
login.clickEmailNext();
login.enterPassword(PASSWORD);
login.clickPasswordNext();
- Стандартный вход через страницу логина
- Последовательное заполнение email и пароля
```

Шаг 2: Поиск непрочитанных писем

```
List<WebElement> unread =
driver.findElements(By.cssSelector("tr.zA.zE"));
if (unread.isEmpty()) {
 return;
}
WebElement first = unread.get(0);
- Поиск всех элементов с классами непрочитанных писем
- Если писем нет - завершение теста
- Выбор первого письма в списке
```

Шаг 3: Изменение статуса письма

```
first.findElement(By.cssSelector("div.oZ-jc")).click();
```

```
driver.findElement(By.cssSelector("li.bqX.brr[data-tooltip='Отметить
как прочитанное']")).click();
```

1. Клик по чекбоксу письма (выделение)
2. Клик по иконке "Отметить как прочитанное" в тулбаре

#### Шаг 4: Снятие выделения

```
first.findElement(By.cssSelector("div.oZ-jc")).click();
```

- Повторный клик по чекбоксу для снятия выделения

#### Технические особенности

1. CSS-классы Gmail:
  - `tr.zA.zE`: Строка таблицы для непрочитанных писем
  - `div.oZ-jc`: Чекбокс выделения письма
  - `li.bqX.brr`: Элемент тулбара с action-кнопками
2. Работа без Page Objects:
  - Прямое использование `driver.findElement()`
  - Отсутствие инкапсуляции логики в отдельные классы
3. Условное выполнение:
  - Корректная обработка случая отсутствия писем
  - Тест не падает при пустом ящике

#### Визуальные изменения после теста

1. Письмо теряет жирное начертание темы
2. Исчезает синий индикатор непрочитанного
3. Письмо перемещается в раздел прочитанных
4. Счётчик непрочитанных уменьшается на 1

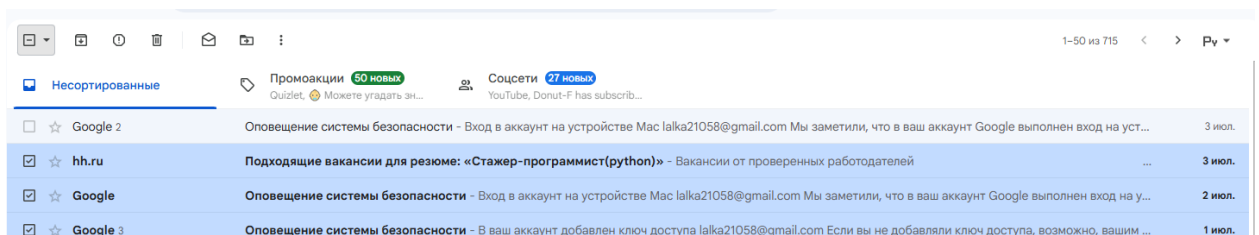


Рисунок 8 - Письма с отметкой

## ТЕСТ (ПОИСК, ПО КЛЮЧЕВЫМ СЛОВАМ)

Тест проверяет функциональность расширенного поиска в Gmail, выполняя последовательно 4 различных сценария поиска. Разберём его работу:

Цель теста

Проверить различные варианты расширенного поиска:

1. Поиск по ключевым словам
2. Поиск с несуществующими запросами
3. Поиск по фразам
4. Граничные случаи ввода
5. Стабильность работы поискового функционала

Пошаговая логика работы

Шаг 1: Авторизация

```
InboxPage inbox = auth(USER_EMAIL, USER_PASSWORD);
```

- Стандартный вход через метод базового класса

Шаг 2: Инициализация поиска

```
AdvancedSearchPage search = new AdvancedSearchPage(driver);
```

- Создание объекта для работы с расширенным поиском

Сценарии поиска (1-4):

```
search.openSearch();
```

```
search.typeFirstQuery(...);
```

```
search.typeSecondQuery(...);
```

```
search.submitSearch();
```

```
search.pauseFiveSeconds();
```

```
search.clearFirst();
```

1. Открытие формы расширенного поиска
2. Ввод запроса в разные поля поиска
3. Запуск поиска
4. Пауза для визуальной проверки/ожидания результатов
5. Очистка предыдущего запроса



## Предполагаемые поля поиска

### 1. Первое поле (typeFirstQuery):

- Вероятно: "От" (отправитель) или "Тема"

```
public void typeFirstQuery(String text) {
 driver.findElement(FIRST_FIELD).sendKeys(text);
}
```

### 2. Второе поле (typeSecondQuery):

- Вероятно: "Содержит слова" или "Текст письма"

```
public void typeSecondQuery(String text) {
 driver.findElement(SECOND_FIELD).sendKeys(text);
}
```

|                          |                        |          |                                                                                                                                                                            |            |
|--------------------------|------------------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| <input type="checkbox"/> | ☆ NVIDIA Accounts      | Входящие | Аутентификация по адресу эл. почты - <a href="#">Здравствуйте!</a> Нажмите ссылку внизу, чтобы подтвердить этот адрес и перейти в NVIDIA GeForce Experienc...              | 2 мар.     |
| <input type="checkbox"/> | ☆ Андрей Павлов        | Входящие | Яндекс.Диск: доступ к папке «3388» - <a href="#">Здравствуйте!</a> Андрей Павлов ( rawlov.andrusha2014@yandex.ru ) предоставляет вам полный доступ к папке...              | 11.09.2023 |
| <input type="checkbox"/> | ☆ Stepik Team          | Входящие | Рекомендуемые курсы на Stepik - <a href="#">Здравствуйте!</a> WHO666H! Мы хотели бы порекомендовать вам некоторые курсы на Stepik. Для оплаты платных...                   | 11 июн.    |
| <input type="checkbox"/> | ☆ QR-Code.io           | Входящие | Добро пожаловать в QR-Code.io - Для получения дополнительной помощи посетите раздел часто задаваемых вопросов или свяжитесь с нами н...                                    | 1 февр.    |
| <input type="checkbox"/> | ☆ NVIDIA Accounts      | Входящие | Подтверждение адреса эл. почты в NVIDIA - <a href="#">Здравствуйте!</a> Вы указали этот адрес электронной почты для входа в учетную запись NVIDIA. Нажмит...               | 16.08.2023 |
| <input type="checkbox"/> | ☆ Группа 3388 каф. МО. | Входящие | Приглашение в группу "Группа 3388 каф. МОЭВМ (ФКТИ)" - <a href="#">Здравствуйте!</a> lalka21058@gmail.com! onextoplay@gmail.com приглашает Вас присоед...                  | 20.09.2023 |
| <input type="checkbox"/> | ☆ Mathway              | Входящие | Добро пожаловать в Mathway - <a href="#">Здравствуйте!</a> Добро пожаловать в Mathway, ваш личный помощник при выполнении домашних заданий по мат...                       | 08.07.2023 |
| <input type="checkbox"/> | ☆ QR-Code.io           | Входящие | [Напоминание] Не забудьте скачать QR-код - <a href="#">Здравствуйте!</a> Ваши друзья в QR-Code.io. QR-Code.io. support@qr-code.io © 2025, QR-Code.io. Вс...                | 4 февр.    |
| <input type="checkbox"/> | ☆ QR-Code.io           | Входящие | RE: [Требуется действие] Ваш деактивированный QR-код мог быть отсканирован! - <a href="#">Здравствуйте!</a> Ваши друзья в QR-Code.io. QR-Code.io. su...                    | 6 мар.     |
| <input type="checkbox"/> | ☆ QR-Code.io           | Входящие | [Требуется действие] Ваш деактивированный QR-код мог быть отсканирован! - <a href="#">Здравствуйте!</a> Ваши друзья в QR-Code.io. QR-Code.io. suppor...                    | 3 мар.     |
| <input type="checkbox"/> | ☆ hh.ru                | Входящие | Спасибо за ваше резюме! - <a href="#">Здравствуйте!</a> Евгений! Ваш отклик на вакансию Сотрудник склада Озон (Шушары) успешно зарегистрирован и на...                     | 30.06.2024 |
| <input type="checkbox"/> | ☆ ASOS                 | Входящие | Спасибо за ваш заказ! - <a href="#">Здравствуйте!</a> Женя! Мы получили твой заказ. ASOS - Открой мир моды онлайн. Заказ размещен. <a href="#">Здравствуйте!</a> Женя! ... | 02.01.2022 |
| <input type="checkbox"/> | ☆ Команда Gmail        | Входящие | В Gmail всё по полочкам - <a href="#">Здравствуйте!</a> Женя! Пользоваться Gmail удобно. Видео о почте. Категории входящих сообщений. Gmail группирует в...                | 24.05.2015 |

## Рисунок 9 - Письма с пометкой "Здравствуйте"

## ТЕСТ (ПРОВКА ВОССТАНОВЛЕНИЯ СЕССИИ ПОСЛЕ РАЗРЫВА СОЕДИНЕНИЯ)

Пояснение теста `OfflineLoginTest`

Этот тест проверяет поведение Gmail при потере интернет-соединения.

Основная цель - убедиться, что:

1. При отключении сети появляется корректное сообщение об ошибке
2. После восстановления соединения работа сервиса восстанавливается

Ключевые шаги теста

1. Подготовка и авторизация

```
OfflinePage offlinePage = new OfflinePage(driver);
offlinePage.enableNetwork(); // Гарантировано включаем сеть
InboxPage inbox = auth(USER_EMAIL, USER_PASSWORD); //
```

Стандартная авторизация

2. Эмуляция отключения интернета

```
offlinePage.emulateOffline(); // Программное отключение сети
offlinePage.refresh(); // Принудительное обновление страницы
```

3. Проверка сообщения об ошибке

```
String errorText = offlinePage.getOfflineErrorText().toLowerCase();
assertTrue(
 errorText.contains("offline") || errorText.contains("internet"),
 "Ожидалось сообщение об offline, но получили: " + errorText
);
- Проверяет наличие ключевых слов в сообщении об ошибке
- Учитывает разные варианты локализации ("offline", "internet",
"сеть" и т.д.)
```

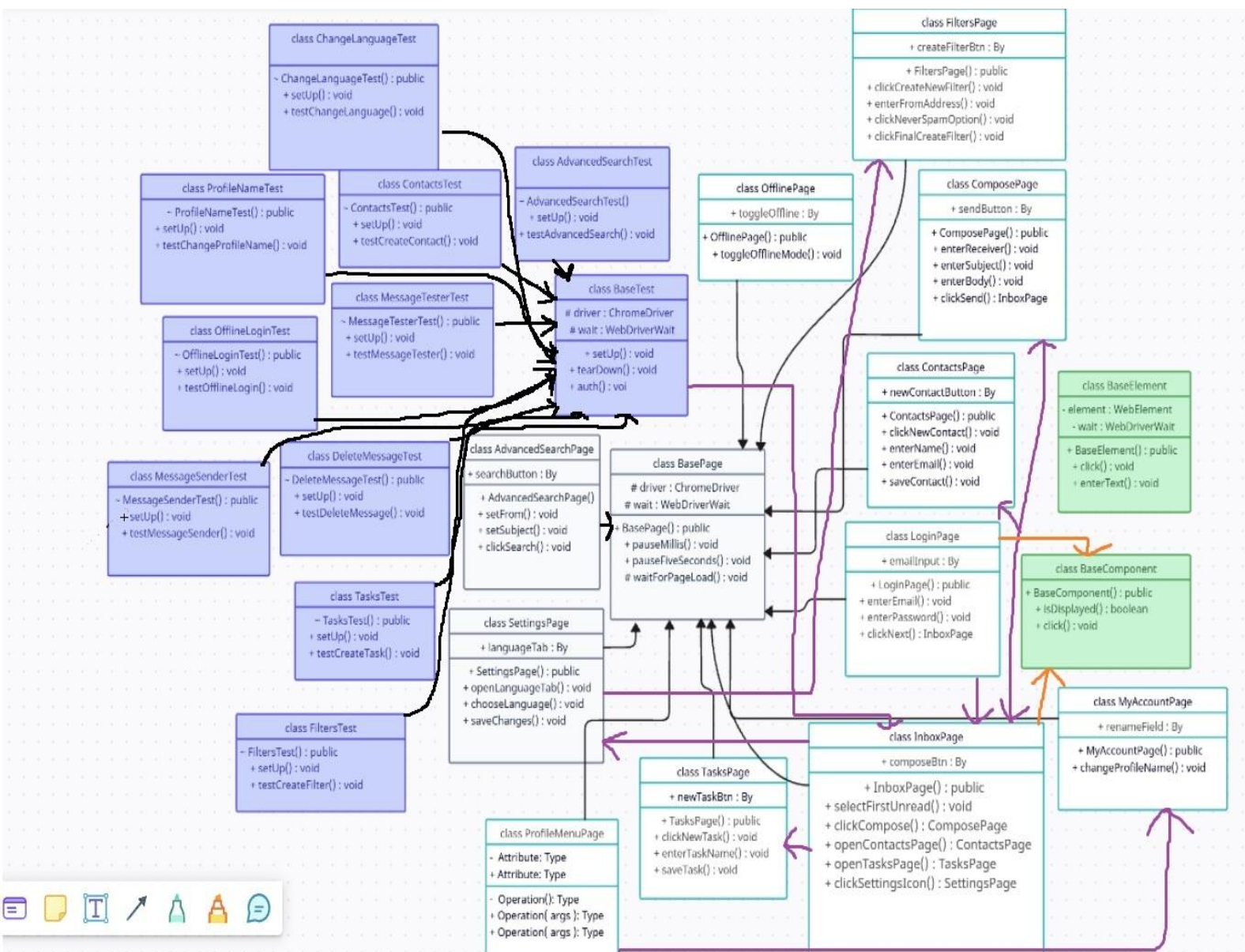
4. Восстановление работы

```
offlinePage.emulateOnline(); // Включаем сеть обратно
```

```
driver.navigate().to(INBOX_URL); // Переход в папку
"Входящие"
wait.until(ExpectedConditions.titleContains("Входящие")); //
```

Ожидание загрузки

## UML – ДИАГРАММА



ЦВЕТОВАЯ КОДИРОВКА И ТИПЫ СВЯЗЕЙ:

- синие блоки – тест-классы (JUnit/тестовые сценарии);
- белые блоки – Page Object-классы (страницы);
- зелёные блоки – базовые/утилитные классы;
- чёрные стрелки – наследование (extends);
- красные стрелки – ассоциации: методы одного класса возвращают объект другого;

- жёлтые стрелки – зависимости (static util-вызовы, использование констант).

## Вывод

Над проектом автоматизации тестирования Gmail мы работали втроем, организовав процесс через GitHub с четким разделением обязанностей. Я отвечал за модуль авторизации и функционал работы с письмами, включая удаление сообщений и пометку прочитанных. Второй участник взял на себя реализацию тестов для настроек — фильтров, смены языка интерфейса и расширенного поиска. Третий участник разрабатывал тесты для работы с контактами, задачами, offline-режимом и отправкой писем.

Для координации мы создали общий репозиторий с единой структурой проекта, где сразу установили ключевые соглашения: именование методов в camelCase, обязательное использование Page Object Model и вынесение стандартных локаторов в BaseComponent. Каждый день мы проводили короткие 15-минутные созвоны для синхронизации прогресса, а для управления кодом использовали Git Flow с обязательным ревью перед мерджем в основную ветку.

В процессе столкнулись с типичными проблемами командной разработки. Конфликты версий решали через систему feature-веток и pull-request'ов. Чтобы предотвратить расхождения в стиле кодирования, внедрили Checkstyle с общим конфигурационным файлом. При обнаружении дублирующихся методов сразу выносили их в BasePage, а проблемы с зависимостями фиксировали через жесткое закрепление версий в pom.xml.

Список используемой литературы:

1. Официальная документация по Java <https://docs.oracle.com/en/java/>
2. Официальная документация по Selenide  
<https://selenide.org/documentation.html>
3. Официальная документация по Maven  
<https://maven.apache.org/guides/index.html>
4. Сайт gmail.com <https://mail.google.com/>