**Вскрытие покажет**

Вот сам проект на гитхабе: https://github.com/selenide-examples/gmail

Давайте рассмотрим некоторые моменты подробнее.

**Настройка**

Перед запуском тестов надо увеличить таймаут, т.к. в GMail часто элементы грузятся дольше 4 секунд, которые прописаны в Selenide по умолчанию. Поставим 10 секунд. Хотя на некоторых конференциях, где интернет слабенький, мне и этого не хватало.

@BeforeClasspublicstaticvoidopenInbox(){timeout=10000;baseUrl="http://gmail.com";open("/");$(byText("Loading")).should(disappear);login();}

Обратили внимание, как мы ждём окончания загрузки страницы? В этом вся мощь Selenide: элемент легко найти по тексту, и легко дождаться, пока он пропадёт.

**Логин**

Логин происходит очень просто:

privatestaticvoidlogin(){$("#Email").val(System.getProperty("gmail.username","enter-your-gmail-username"));$("#Passwd").val(System.getProperty("gmail.password","enter-your-gmail-password"));$("#signIn").click();$(".error-msg").waitUntil(disappears,2000);}

Последняя строчка нужна для того, чтобы тест быстро упал, если вы ввели неверный пароль.

**Количество непрочитанных писем**

В реальной жизни я бы запускал тесты с определённым набором тестовых данных со, скажем, 4 непрочитанными письмами. И тогда я бы в тесте проверил наличие надписи “Inbox (4)”:

@TestpublicvoidshowsNumberOfUnreadMessages(){$(By.xpath("//div[@role='navigation']")).find(withText("Inbox (4)")).shouldBe(visible);}

(Но тут у нас случай сложнее, нам приходится тестировать с реальными данными, которые непостоянны. Поэтому ограничимся просто проверкой наличия слова “Inbox”.)

**Проверка содержимого инбокса**

Увы, у GMail все локаторы тщательно заобфускированы. Использовать ID или другие селекторы не представляется возможным. Поэтому ограничимся просто проверкой наличия на странице определённых текстов - мы знаем, что такие письма точно должны быть в моём инбоксе:

@TestpublicvoidinboxShowsUnreadMessages(){$$(byText("Gmail Team")).filter(visible).shouldHave(size(1));$$(byText("LastPass")).filter(visible).shouldHave(size(3));$$(byText("Pivotal Tracker")).filter(visible).shouldHave(size(3));}

**Обновление инбокса**

В интерфейсе GMail есть кнопка обновления “Refresh”. Ищем её по атрибуту title=Refresh - другого способа я не нашёл.

@TestpublicvoiduserCanRefreshMessages(){// В реальной жизни: INSERT INTO messages ...$(by("title","Refresh")).click();// В реальной жизни: проверить, что новое письмо появилось в инбоксе}

В настоящих тестах я до нажатия добавил бы новое письмо в базу данных, и после нажатия “Refresh” проверил бы, что оно появилось в инбоксе.

**Новое письмо**

Для составления нового письма нажимаем кнопку “COMPOSE”:

$(byText("COMPOSE")).click();

И вбиваем адрес, тему и текст:

$(By.name("to")).val("andrei.solntsev@gmail.com").pressTab();$(by("placeholder","Subject")).val("ConfetQA demo!").pressTab();$(".editable").val("Hello braza!").pressEnter();$(byText("Send")).click();

В общем-то, совсем ничего сложного.

И в конце проверяем, что письмо отослалось:

$(withText("Your message has been sent.")).shouldBe(visible);

**Undo - redo**

В моём почтовом ящике включена экспериментальная фича - “Undo”. После отсылки письма в течение 10 секунд горит кнопка “Undo”, нажав на которую, я могу отменить отсылку последнего письма и вернуться к редактированию. Очень удобно, когда, нажав “Send”, обнаруживаешь, что случайно послал письмо не тому человеку.

Итак, попробуем это протестировать. В конце предыдущего теста добавляем:

$(byText("Undo")).click();highlight($(byText("Sending has been undone.")).should(appear));

Подправляем текст письма:

$(".editable").should(appear).append("Hello from ConfetQA Selen").pressEnter().pressEnter();$(byText("Send")).click();

И ждём 10 секунд, пока кнопка “Undo” пропадёт:

highlight($(withText("Your message has been sent.")).should(appear));highlight($(byText("Undo")).should(appear)).waitUntil(disappears,12000);$(byText("Sent Mail")).click();

Письмо окончательно отослано адресату.

Вот теперь GMail протестирован, гугл может спать спокойно.

**Выводы**

Как видите, интерфейсы без хороших ID тоже можно тестировать. Долгие запросы и ajax не являются помехой. Динамический контент не препятствие. Вот простые рецепты, позволяющие бороться с этими трудностями:

* *Поиск элементов по тексту.*  
  Это максимально приближено к поведению пользователя и минимально зависит от реализации веб-приложения.
* *Тестировать только важное*  
  В приложении GMail ещё куча вещей осталась непротестированными. В данных условиях их, кажется, и невозможно протестировать (невозможность подсунуть тестовые данные, отсутствие статических локаторов). Но это не так уж и важно - главное, что критичные функции мы протестировали: показ писем в инбоксе и отсылка нового письма. Всегда начинайте с тестирования критичной функциональности, а остальное успеете оценить позже.
* *Используйте правильные инструменты*  
  Selenide автоматически решает проблемы с таймаутами, аяксом, динамическим контентом, поиском по тексту. Тесты не загрязнены длинными страшными XPath. Тесты содержат только логику. Тесты хорошо читаемые.