ФГБОУ ВО «ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И ИНФОМРАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра информатики и математического обеспечения

Отчет о тестировании приложения

«Retrospectify»

Выполнил:
Лайтинен Н.В.
22608

Петрозаводск 2018

План тестирования

1.Объект тестирования

Объектом тестирования выбрано веб-приложение «Retrospectify».

Приложение имеет вид веб-сайта (рисунок 1).

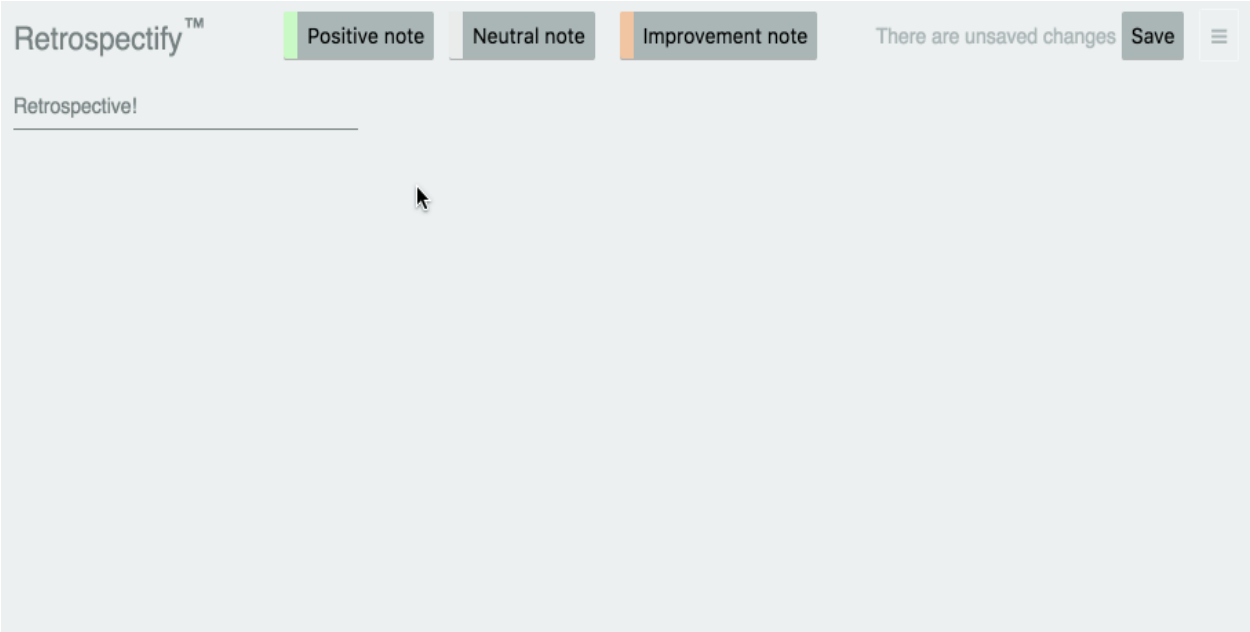

Рисунок 1 – интерфейс «Retrospectify»

Написано с использованием javascript фреймворка Vue.js.

Основная идея приложения – создание трех видов заметок - задач:

1. Текущие
2. Важные
3. Требуют правок

При работе с приложением пользователь имеет возможность:

1. Создавать заметки;
2. Редактировать заметки:
    a. Устанавливать размер шрифта;
    b. Устанавливать приоритет;
    c. Менять размер.
3. Удалять заметки;
4. Перемещать заметки по рабочей области;
5. Упорядочивать заметки по степени приоритета;
6. Создавать разные наборы досок со своим списком заметок;
7. Менять название рабочей области;

8. Сохранять/удалять рабочую область;
9. Экспортировать все заметки текущей рабочей области в файл.

Тестирование заключается в виде тестирования компонентов веб-приложения на клиентской стороне.

В **аттестационном** тестировании участвует следующий функционал веб-сервиса:

- создание заметок;
- перемещение заметок;
- создание нескольких рабочих областей;
- изменение размера заметки;
- экспорт в файл.

2.Стратегия тестирования

2.1 Архитектура веб-приложения

Веб-приложение Retrospectify имеет компонентную архитектуру (рисунок2).

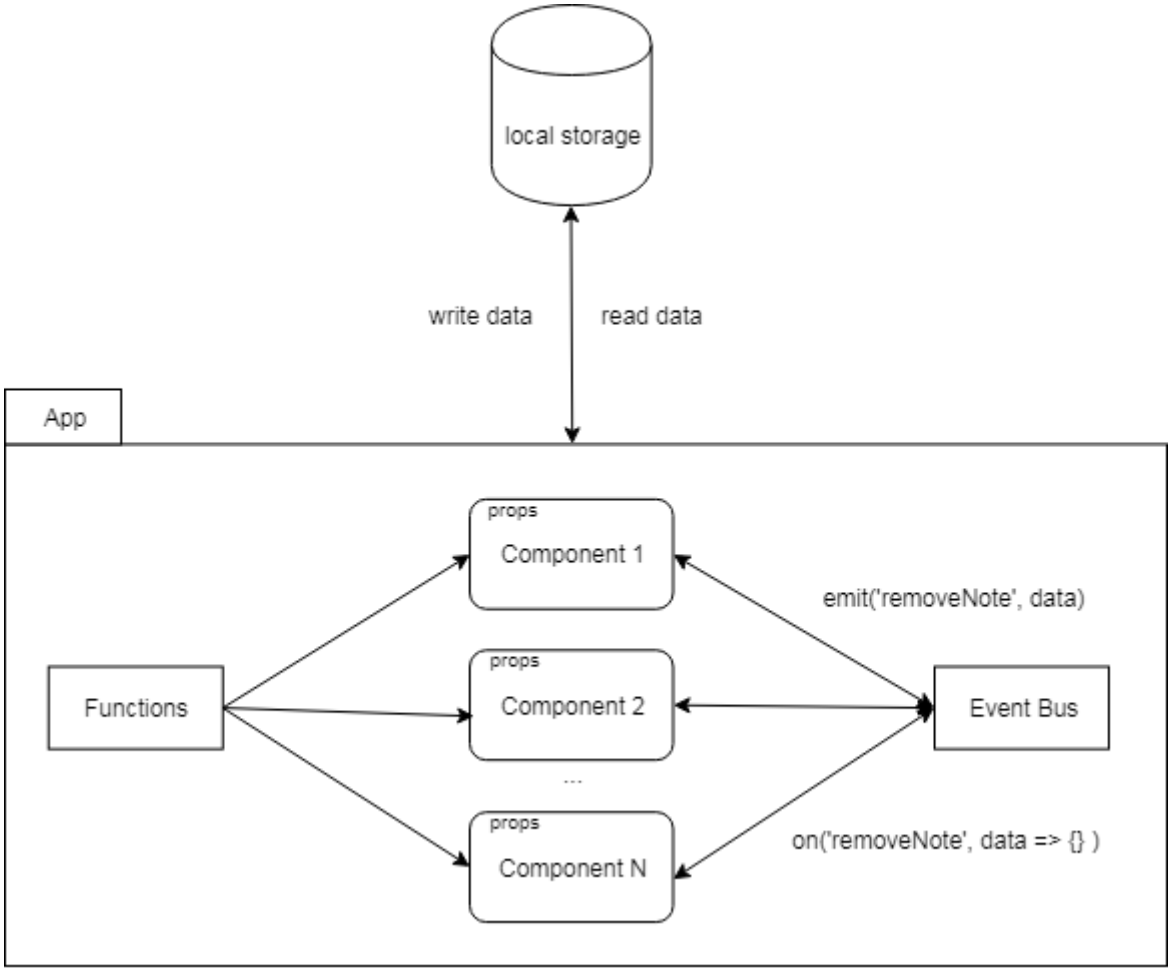

Рисунок 2 – архитектура «Retrospectify»

Взаимодействие компонентов между собой происходит при помощи централизованной шины событий - Event Bus. Хранение данных реализовано при помощи LocalStorage. Для передачи данных между компонентами родитель-потомок используется подход Props Down Events Up.

2.2 Компоненты и тестирование

Веб-приложение Retrospectify состоит из 5 основных компонентов. Каждый компонент содержит некоторое количество методов. Далее каждый метод компонентов описан с позиции: что он делает, какие входные параметры получает, что возвращает и какие у этого метода зависимости от других функций. Указание зависимостей от других функций необходимо для того, чтобы при написании тестов создать соответствующие mock объекты.

App

Компонент App является корневым компонентом веб-приложения. Инициализирует приложение.

Включает в себя компоненты:

- Board;
- SavedBoards.

Использует модули:

- BoardExport;
- Event Bus.

Методы, **НЕ** участвующие в unit тестировании:

- resetActive () - отправляет всем слушателям сигнал 'reset-active';
- saveBoards () - отправляет всем слушателям сигнал 'save-boards';
- toggleSidebar () - отправляет всем слушателям сигнал 'toggle-sidebar'.

Основные методы для unit тестирования:

- data():

  Функция инициализации компонента глобальными данными.

  Входные параметры: нет

  Выходные параметры:

  - activeBoardIndex: 0 (число) – номер активной доски;
  - unsavedChanges: false (boolean) – сохранена/не сохранена текущая сессия;
  - boards: [] – массив досок, хранящий объекты.

  Зависимости: отсутствуют.

- activeBoard():

  Функция получения активной доски.

  Входные параметры (косвенные):

  - boards – массив [] досок;

- activeBoardIndex – (число) номер активной доски;

Выходные параметры: объект доски {…} или false.

Зависимости: отсутствуют.

- boardTitle ():

  Функция генерирования названия доски.

  Входные параметры (косвенные):

  - date – текущая дата в формате “Декабрь 3, 2018”

  Выходные параметры

  - строка - `My retrospective for ${date}`, где ${date} – входной параметр

  Зависимости:

  - time() – функция, возвращающая дату в указанном формате, возвращает строку.

- createBoard (initial):

  Функция создания объекта доска.

  Входные параметры :

  - initial – (boolean) значения.

  Выходные параметры:

  - board – объект {…} доска, с полями:
    - title: (string) - строка, название доски;
    - notes: [] – массив заметок;
    - initial: initial (boolean) – флаг инициализации.

  Зависимости: boardTitle().

- loadState ():

  Функция загрузки данных из LocalStorage.

  Входные параметры (косвенные):

  - window.localstorage {…} – объект с полями, которые хранят данные о досках, содержат функции для извлечения данных, установки данных.

  Выходные параметры:

  - boards [] – массив досок, каждый элемент которого является объектом доска {…}.

  Зависимости: window.localstorage, migrateState()

- saveState ():

  Функция загрузки данных в localStorage и обновления информации о проведенных изменениях.

  Входные параметры (косвенные):

- window.localstorage {…} – объект с полями, которые хранят данные о досках, содержат функции для извлечения данных, установки данных;
- boards [] – массив досок, каждый элемент которого является объектом доска {…}.

Выходные параметры:

- content {…} – объект со списком досок, загруженных в localstorage;
- unsavedChanges = false - не сохраненные данные отсутствуют.

Зависимости: window.localstorage.

- migrateState ():

  Функция миграции старой версии данных из localStorage в новую версию с последующей записью этих данных обратно в хранилище. Также проведение обновления версии данных, записанных в хранилище.

  Входные параметры (косвенные):

  - window.localstorage {…} – объект с полями, которые хранят данные о досках, содержат функции для извлечения данных, установки данных;
  - boards [] – массив досок, каждый элемент которого является объектом доска {…}.

  Выходные параметры:

  - content {…} – объект со списком досок, загруженных в localstorage;
  - VERSION – число, версия данных.

  Зависимости: window.localstorage.

- created ():

  Функция-хук, которая срабатывает после создания экземпляра приложения, вызывает функцию loadState(), а также createBoard(true), если массив boards[] пустой.

  Входные параметры (косвенные):

  - boards [] – массив досок, , каждый элемент которого является объектом доска {…}.

  Выходные параметры: отсутствуют

  Зависимости: loadState(), createBoard().

Компонент Board

Компонент Board является дочерним компонентом веб-приложения. Отвечает за манипулирование заметок по рабочей области.

Включает в себя компоненты:

- Note;

Использует модули:

- Positioner;
- Event Bus.

Методы, **НЕ** участвующие в unit тестировании:

- Data() - функция инициализации начального состояния компонента;
- stopDrag (id) - отправляет всем слушателям сигнал 'stop drag, id = id';
- getNoteById (id) - функция получения объекта заметки;
- getMaxOrder () - функция получения максимального числа степени важности среди всех заметок.

Основные методы для unit тестирования:

- startDrag (id):

  Callback функция, которая срабатывает, когда происходит процесс перемещения заметки по рабочей области.

  Входные параметры:

  - id – (число) – номер заметки.

  Выходные параметры:

  - activeDrag: id (число) – номер активной заметки.

  Зависимости:

  - getMaxOrder() – функция получения максимального числа степени важности среди всех заметок. Возвращает число от 0 и выше;
  - getNoteById(id) – функция получения объекта заметки. Возвращает объект заметки с полями;
  - updateNote(id, {order}) –функция обновления данных заметки.

- updateNote (id, update):

  Функция обновления данных заметки.

  Входные параметры + (косвенные):

  - id – (число) – номер заметки;
  - update – (объект) – данные для обновления;
  - board – (объект) – доска с данными о заметках.

Выходные параметры: объект заметки {…} обновленный.

Зависимости:

- getNoteById(id) - функция получения объекта заметки. Возвращает объект заметки с полями.

- addNote (type):

Функция создания новой заметки.

Входные параметры + (косвенные):

- type – (строка) – тип заметки (improvement, neutral, positive);
- board – (массив) – массив объектов заметок.

Выходные параметры : объект заметки с полями:

- text – (строка) – текст заметки;
- note_type – (строка) – тип заметки;
- position - (объект) – координаты x y заметки на холсте;
- noteSize - (объект) – ширина высота заметки;
- fontSize – (число) – размер шрифта заметки;
- votes – (число) – важность заметки;
- order – (число) – максимальная важность заметок + 1;
- id – (число) – уникальное имя заметки.

Зависимости:

- getMaxOrder() - функция получения максимального числа степени важности среди всех заметок. Возвращает число от 0 и выше;
- getPositionforNew() – функция получения координат x y для новой заметки на холсте.

Компонент Note

Компонент Note является дочерним компонентом веб-приложения. Получает данные от компонента Board. Отвечает за редактирование заметки.

Использует модули:

- Event Bus.

Методы, **НЕ** участвующие в unit тестировании:

- Data() - функция инициализации начального состояния компонента;
- removeNote () - функция отправляет всем слушателям сигнал "remove-note', this.id';
- decFontSize () - функция уменьшения размера шрифта заметки;
- addVote () - функция отправляет всем слушателям сигнал "update', this.id, {votes: this.votes + 1}';
- removeVote - функция отправляет всем слушателям сигнал "update', this.id, {votes: this.votes - 1}';
- onPositionMouseMoveStart() - функция отправляет всем слушателям сигнал "start-drag', this.id';
- onPositionMouseMoveStop () - функция отправляет всем слушателям сигнал "stop-drag', this.id'.

Основные методы для unit тестирования:

- incrFontSize ():

  Функция увеличения размера шрифта заметки.

  Входные параметры (косвенные):

  - fontSize – (число) – размер шрифта заметки.

  Выходные параметры:

  - newFontSize - (число) – новый размер шрифта заметки.

  Зависимости: -

- onPositionMouseMove (d):

  Callback функция, которая срабатывает, когда происходит перемещение заметки мышкой по холсту. Рассчитывает новые координаты и оповещает все компоненты, что событие произошло и передает обновленные координаты заметки на холсте.

  Входные параметры + (косвенные):

  - d – (объект) – координаты x и y заметки на холсте;
  - Dragging – (boolean) – флаг, происходит или нет перемещение заметки.

  Выходные параметры:

  - newD- (объект) – новые координаты x и y заметки на холсте;

- id – (число) – id заметки.

Зависимости: -

- onSizeMouseMove (d):

  Callback функция, которая срабатывает, когда происходит изменение размера заметки мышкой. Рассчитывает размеры и повещает все компоненты, что событие произошло и передает обновленные размеры ширины и высоты заметки.

  Входные параметры + (косвенные):

  - d – (объект) – размер x y изменения ширины высоты заметки;
  - noteSize – (объект) – текущий размер заметки.

  Выходные параметры:

  - noteSize - (объект) – новый размер заметки;
  - id – (число) – id заметки.

  Зависимости: -

Компонент SavedBoards

Данный компонент отвечает только за ui отрисовку веб приложения. Вся логика заключается в оповещении остальных компонентов событиями и передачи им данных.

НЕ тестируется.

Основные методы:

- loadBoard (id) - отправляет всем слушателям сигнал "load-board', id';
- createBoard () - отправляет всем слушателям сигнал 'create-board';
- removeBoard (id) - отправляет всем слушателям сигнал "remove-board', id';
- clearBoard () - отправляет всем слушателям сигнал "clear-board";
- saveBoards () - отправляет всем слушателям сигнал "save-boards";
- exportBoard (id) - отправляет всем слушателям сигнал ' 'export-board', id'.

Модуль Positioner

Данный модуль реализует функционал позиционирования заметки на холсте.

Нормализует размеры заметки, определяет наиболее подходящие позиции на холсте.

Методы, **НЕ** участвующие в unit тестировании:

- normalizeDimensions (note) – функция нормализации размеров заметки;
- reArrange () – функция сброса важности заметок.

Основные методы для unit тестирования:

- getPositionforNew(terciary):

  Функция определения наиболее удачной позиции x y создаваемой заметки.

  Входные параметры (косвенные):

  - terciary – (число) – 0, 1 или 2 в зависимости от типа заметки.

  Выходные параметры:

  - position - (объект) – позиция x y заметки.

  Зависимости: window.innerWidth – размер рабочей области.

2.3 Стратегии тестирования

Для тестирования javascript проекта Retrospectify используется Vue Test Utils.
Vue Test Utils — официальная библиотека модульного тестирования для Vue.js.
Jest – фреймворк, разработанный Facebook для тестирования кода JavaScript .
Используется в связке с Vue Test Utils для запуска тестов.

Стратегия блочного тестирования

В блочном тестировании участвую указанные ранее методы в описании компонентов веб-сервиса. Блочное тестирование реализуется при помощи Vue Test Utils и Jest.

Стратегия интеграционного тестирования

Для проведения интеграционного тестирования выделены следующие методы:

- createBoard() + boardTitle();
- loadState() + migrateState();
- created() + loadState();
- created() + CreateBoard();
- startDrag() + updateNote();
- addNote() + getPositionForNew().

Интеграционное тестирование будет проводиться по схеме “сверху-вниз”. Т.е. вначале будут протестированы модули верхнего уровня, а их зависимости будут заменены mock-объектами. Далее постепенно mock-объекты будут заменяться на реальные модули. Данная схема выбрана по причине того, что предполагается расширение объекта тестирования и не все спроектированные интерфейсы имеют конкретную реализацию. Схема интеграции представлена на рис.3

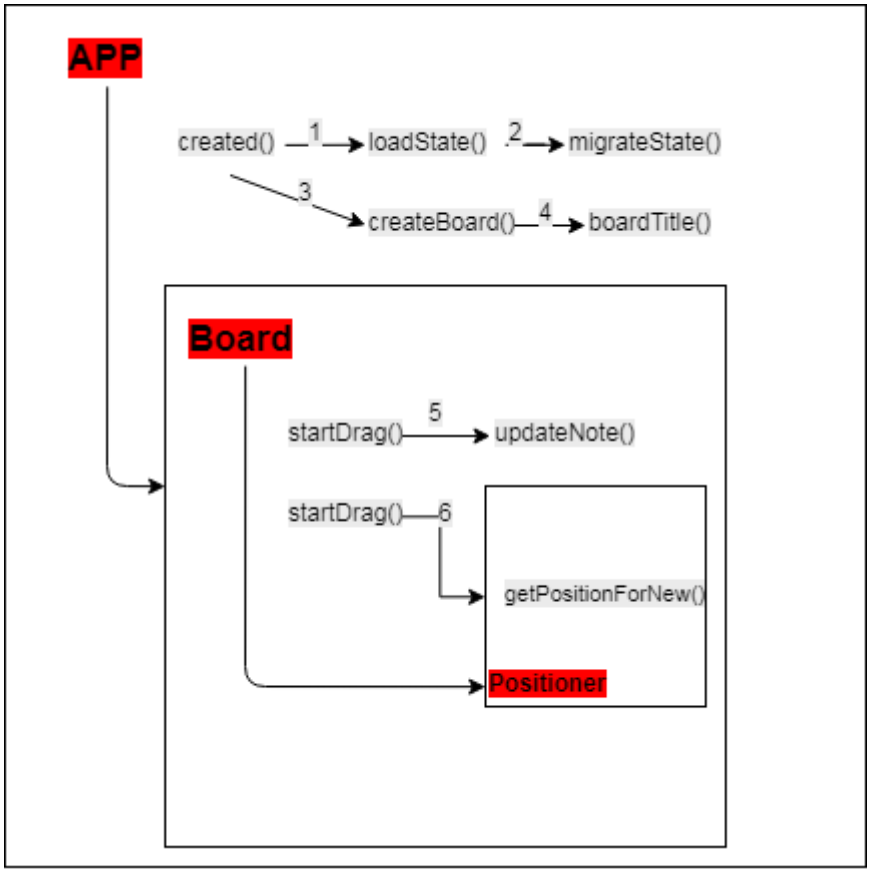

Рисунок 3 – схема интеграции

Каждый этап интеграции содержит пары функций, для взаимодействия которых далее будет составлен детальный план теста.

Этапы интеграции:

- Модуль APP:
  1) Вызов ф-и created(), внутри которой вызов ф-и loadState();
  2) Вызов ф-и loadState(), внутри которой вызов ф-и migrateState();
  3) Вызов ф-и created(), внутри которой вызов ф-и createBoard();
  4) Вызов ф-и createBoard(), внутри которой вызов ф-и boardTitle ().
- Модули APP, Board:
  5) Вызов ф-и startDrag(), внутри которой вызов ф-и updateNote();
- Модули APP, Board, Positioner:
  6) Вызов ф-и addNote(), внутри которой вызов ф-и getPositionForNew();

Стратегия аттестационного тестирования

В **аттестационном** тестировании участвует следующий функционал веб-сервиса:

- создание заметок;
- перемещение заметок;

- создание нескольких рабочих областей;
- сохранение изменений;
- экспорт в файл.

Аттестационное тестирование будет проводится методом «Живого человека». В роли такого человека выступает сам автор тестирования.

Тестирующий человек, по заранее заданным инструкциям (Test Cases), производит требуемые действия и сверяется с заранее заданными результатами. Тест считается пройденным если в результате полученные такие же данные, которые описаны в инструкции. В противном случае тест считается негативным.

Стратегия нагрузочного тестирования

Веб-сервис предполагает создание неограниченного количества досок с неограниченным количеством заметок.

В рамках нагрузочного тестирования проводится процесс создания досок в количестве от 5-15, на каждой из которых по 50-100 заметок. Каждая заметка располагается в случайном порядке. После создания заметок, открывается от 5 до 15 вкладок браузера и проверяется синхронизация данных. Затем браузер перезапускается и открывается одна вкладка с сервисом и проверяется наличие сохраненных данных.

Критерий прохождения тестов

Тест считается успешно пройденным, если ожидаемый и фактический результаты совпадают. Если тест завершается неудачей, то перед принятием решения целесообразно проверить правильность самого теста. Если тест завершился неудачей и тест реализован правильно, то производится заключение о найденной ошибке. Тестирование считается пройденным, если во время его прохождения не выявлено критических ошибок, а процент непройденных тестов меньше 1% от общего количества.

Критерий приостановки тестов

Тестирование должно быть приостановлено, если количество непройденных тестов превысит 10% от их общего количества. Тестирование должно быть приостановлено при обнаружении критических ошибок.

Критерий возобновления тестирования

Тестирование возобновляется после исправления ошибок, выявленных при предыдущем тестировании.

3 Детальный план тестов

3.1 Блочные тесты

Компонент App

№	Б1
Цель теста:	Проверка работоспособности функции data()
Метод:	Черный ящик
Тип:	Позитивный
Объект теста:	Функция data()
Входные данные:	-
Ожидаемый результат:	Возвращает объект data = {}, с полями: activeBoardIndex: 0, unsavedChanges: false, boards: []

№	Б2
Цель теста:	Проверка работоспособности функции activeBoard ()
Метод:	Черный ящик
Тип:	Позитивный
Объект теста:	Функция activeBoard ()
Входные данные:	Boards = []
Ожидаемый результат:	false

№	Б3
Цель теста:	Проверка работоспособности функции createBoard (initial)
Метод:	Черный ящик
Тип:	Позитивный
Объект теста:	Функция createBoard (initial)
Входные данные:	True, mock boardTitle() = ‘My title’
Ожидаемый результат:	Board = { title = ‘My title’, notes: [], initial: true }

№	Б4
Цель теста:	Проверка работоспособности функции loadState ()
Метод:	Черный ящик
Тип:	Позитивный
Объект теста:	Функция loadState ()

Входные данные:	mock window.localstorage.setItem(‘data’ : { boards: [] })
Ожидаемый результат:	Boards = []

№	Б5
Цель теста:	Проверка работоспособности функции saveState ()
Метод:	Черный ящик
Тип:	Позитивный
Объект теста:	Функция saveState ()
Входные данные:	mock window.localstorage.setItem(‘data’ : { boards: [] }) boards []
Ожидаемый результат:	unsavedChanges = false, mock window.localstorage.getData(‘data) = boards []

№	Б6
Цель теста:	Проверка работоспособности функции migrateState ()
Метод:	Черный ящик
Тип:	Позитивный
Объект теста:	Функция migrateState ()
Входные данные:	mock window.localstorage.setItem(‘data’ : { boards: [] })
Ожидаемый результат:	window.localstorage.getData(‘data) = boards []

№	Б7
Цель теста:	Проверка работоспособности функции created ()
Метод:	Черный ящик
Тип:	Позитивный
Объект теста:	Функция created ()
Входные данные:	boards [], mock loadState(), createBoard(true)
Ожидаемый результат:	Mock функции вызваны по одному разу

№	Б8
Цель теста:	Проверка работы функции при некорректном типе данных
Метод:	Белый ящик
Тип:	Негативный

Объект теста:	Функция createBoard (initial)
Входные данные:	Initial = ‘Hello World!’
Ожидаемый результат:	Выброс исключения “The type of arg must be Boolean!”

№	Б9
Цель теста:	Проверка работы функции при отсутствии параметра
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция createBoard (initial)
Входные данные:	Initial = ‘’
Ожидаемый результат:	Выброс исключения ‘The argument isn’t found’

№	Б10
Цель теста:	Проверка работы функции при невозможности работы с localstorage
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция saveState ()
Входные данные:	mock window.localstorage = undefined boards []
Ожидаемый результат:	Выброс исключения ‘The LocalStorage isn’t found’

№	Б11
Цель теста:	Проверка работы функции при невозможности работы с localstorage
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция loadState ()
Входные данные:	mock window.localstorage = undefined
Ожидаемый результат:	Выброс исключения ‘The LocalStorage isn’t found’

№	Б12
Цель теста:	Проверка работы функции при невозможности работы с localstorage
Метод:	Белый ящик
Тип:	Негативный

Объект теста:	Функция migrateState ()
Входные данные:	mock window.localstorage = undefined
Ожидаемый результат:	Выброс исключения ‘The LocalStorage isn’t found’

№	Б13
Цель теста:	Проверка отработки условия IF
Метод:	Белый ящик
Тип:	Тестирование ветвей
Объект теста:	Функция migrateState ()
Входные данные:	mock window.localstorage.oldState = [{ title = ‘My title’, notes: [‘first’], initial: true }] boards = []
Ожидаемый результат:	mock window.localstorage.state = [{ title = ‘My title’, notes: [‘first’], initial: true }]

Компонент Board

№	Б14
Цель теста:	Проверка отработки условия IF
Метод:	Белый ящик
Тип:	Тестирование ветвей
Объект теста:	Функция startDrag (id)
Входные данные:	Id = 5 mock getMaxOrder() = 4; mock getNoteById(id) = { title = ‘My title’, notes: [‘first’], initial: true, order = 2 };
Ожидаемый результат:	Вызов updateNote(5, {order: 6})

№	Б15
Цель теста:	Проверка отработки условия IF
Метод:	Белый ящик
Тип:	Тестирование ветвей
Объект теста:	Функция startDrag (id)
Входные данные:	Id = 5 mock getMaxOrder() = 4;

	mock getNoteById(id) = { title = ‘My title’, notes: [‘first’], initial: true, order = 4 };
Ожидаемый результат:	Ничего не возвращает

№	Б16
Цель теста:	Проверка работы функции при некорректном типе данных
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция startDrag (id)
Входные данные:	Id = ‘Hello’
Ожидаемый результат:	Выброс исключения “The type of arg must be integer!”

№	Б17
Цель теста:	Проверка работы функции при отсутствии параметра
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция startDrag (id)
Входные данные:	-
Ожидаемый результат:	Выброс исключения ‘The argument isn’t found’

№	Б18
Цель теста:	Проверка работы функции при некорректном типе данных
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	updateNote (id, update)
Входные данные:	Id=’hello’, update=[]
Ожидаемый результат:	Выброс исключения “The type of args isn’t correct!”

№	Б19
Цель теста:	Проверка работы функции при отсутствии параметра
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	updateNote (id, update)
Входные данные:	-

Ожидаемый результат:	Выброс исключения ‘The arguments aren’t found’

№	Б20
Цель теста:	Проверка отработки условия IF
Метод:	Белый ящик
Тип:	Тестирование ветвей
Объект теста:	updateNote (id, update)
Входные данные:	getNoteById(4) = undefined
Ожидаемый результат:	Выброс исключения ‘'Where\'s the note!?’

№	Б21
Цель теста:	Проверка работы функции при некорректном типе данных
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	addNote (type)
Входные данные:	Id=25
Ожидаемый результат:	Выброс исключения “The type of args isn’t correct!”

№	Б22
Цель теста:	Проверка работы функции при отсутствии параметра
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	addNote (type)
Входные данные:	-
Ожидаемый результат:	Выброс исключения ‘The argument isn’t found’

№	Б23
Цель теста:	Проверка отработки switch/case
Метод:	Белый ящик
Тип:	Тестирование ветвей
Объект теста:	addNote (type)
Входные данные:	Type = ‘something’
Ожидаемый результат:	Выброс исключения ‘The type of note isn’t correct!’

Компонент Note

№	Б24
Цель теста:	Проверка отработки условия IF
Метод:	Белый ящик
Тип:	Тестирование ветвей
Объект теста:	incrFontSize()
Входные данные:	fontSize = 100
Ожидаемый результат:	newFontSize = 2.5

№	Б25
Цель теста:	Проверка работы функции при некорректном типе данных
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция onPositionMouseMove (d)
Входные данные:	Id = ‘Hello’
Ожидаемый результат:	Выброс исключения “The type of arg must be obj!”

№	Б26
Цель теста:	Проверка работы функции при отсутствии параметра
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция onPositionMouseMove (d)
Входные данные:	-
Ожидаемый результат:	Выброс исключения ‘The argument isn’t found’

№	Б27
Цель теста:	Проверка отработки условия IF
Метод:	Белый ящик
Тип:	Тестирование ветвей
Объект теста:	Функция onPositionMouseMove (d)
Входные данные:	D = {dx: 5, dy: 0}
Ожидаемый результат:	newD = {x: old + dx, y: old+dy}, old – старые позиции dragging = true

№	Б28
Цель теста:	Проверка отработки условия IF

Метод:	Белый ящик
Тип:	Тестирование ветвей
Объект теста:	Функция onPositionMouseMove (d)
Входные данные:	D = {dx: 0, dy: 0}
Ожидаемый результат:	dragging = false функция ничего прямо не возвращает

№	Б29
Цель теста:	Проверка работы функции при некорректном типе данных
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция onSizeMouseMove (d)
Входные данные:	Id = ‘Hello’
Ожидаемый результат:	Выброс исключения “The type of arg must be obj!”

№	Б30
Цель теста:	Проверка работы функции при отсутствии параметра
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция onSizeMouseMove (d)
Входные данные:	-
Ожидаемый результат:	Выброс исключения ‘The argument isn’t found’

№	Б31
Цель теста:	Проверка отработки условия IF
Метод:	Белый ящик
Тип:	Тестирование ветвей
Объект теста:	Функция onSizeMouseMove (d)
Входные данные:	D = { dx: 1000, dy: 1000}
Ожидаемый результат:	noteSize = {w: 100, h: 50}

Модуль Positioner

№	Б32
Цель теста:	Проверка работы функции при некорректном типе данных

Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция getPositionforNew(id)
Входные данные:	Id = ‘Hello’
Ожидаемый результат:	Выброс исключения “The type of arg must be integer!”

№	Б33
Цель теста:	Проверка работы функции при отсутствии параметра
Метод:	Белый ящик
Тип:	Негативный
Объект теста:	Функция getPositionforNew()
Входные данные:	-
Ожидаемый результат:	Выброс исключения ‘The argument isn’t found’

№	Б34
Цель теста:	Проверка типа возвращаемых значений
Метод:	Черный ящик
Тип:	Позитивный
Объект теста:	Функция getPositionforNew(id)
Входные данные:	2
Ожидаемый результат:	{100, 40} - obj

3.2Интеграционные тесты

№	И1
Цель теста:	Проверка инициализации корневого компонента APP приложения данными
Методы:	Created(), loadState()
Тип:	Позитивный
Объект теста:	Модуль APP
Входные данные:	-
Ожидаемый результат:	Функции отработали. Вернули boards – массив с данными

№	И2
Цель теста:	Проверка возможности миграции данных из хранилища при загрузке данных
Методы:	loadState(), migrateState()
Тип:	Позитивный
Объект теста:	Модуль APP
Входные данные:	mock window.localstorage.setItem(‘data’ : { boards: [] })
Ожидаемый результат:	Функции отработали. Вернули boards – массив с данными []

№	И3
Цель теста:	Проверка возможности создания новой доски при инициализации приложения, при условии, что данных в хранилище нет
Методы:	Created(), createBoard(initial)
Тип:	Позитивный
Объект теста:	Модуль APP
Входные данные:	Initial - boolean
Ожидаемый результат:	Функции отработали. Создан объект obj {}

№	И4
Цель теста:	Проверка генерации названия борда, при его создании
Методы:	createBoard(initial), boardTitle()
Тип:	Позитивный
Объект теста:	Модуль APP
Входные данные:	Initial - boolean
Ожидаемый результат:	Созданный объект борда, расширен полем – название борда – типа string

№	И5
Цель теста:	Проверка обновления данных заметки при ее перемещении по холсту
Методы:	startDrag(id), updateNote (id, update)
Тип:	Позитивный
Объект теста:	Модули APP, Board
Входные данные:	Id - число update – массив

	mock getMaxOrder() = 4;
Ожидаемый результат:	При перемещении данные обновились

№	И6
Цель теста:	Проверка создания заметки и генерации позиции для нее
Методы:	addNote(type), getPositionForNew()
Тип:	Позитивный
Объект теста:	Модули APP, Board
Входные данные:	Type – строка, id - число
Ожидаемый результат:	Объект заметки создан, расширен данными позиции - объект

3.3Аттестационные тесты

№	А1
Тип	Общий
Описание	Используя кнопки интерфейса создать заметку
Начальное состояние	Открытая страница веб приложения
Сценарий аттестационного тестирования	1. Нажать на кнопку «Positive Note» 2. Результат – заметка отобразиться на экране слева 3. Нажать на кнопку «Neutral Note» 4. Результат – заметка отобразиться на экране ближе к центру
Ожидаемый результат	Заметки созданы, они пустые, слева зеленая, справа серо-белая

№	А2
Тип	Общий
Описание	Используя мышку переместить заметку в нижний левый угол холста
Начальное состояние	Открытая страница веб приложения

Сценарий аттестационного тестирования	1. Нажать на кнопку «Positive Note» 2. Результат – заметка отобразиться на экране слева 3. Нажать на заметку мышкой и переместить в нужную позицию 4. Результат-Заметка расположена в задуманной позиции
Ожидаемый результат	Заметка перемещена

№	A3
Тип	Общий
Описание	Используя мышку изменить размер заметки
Начальное состояние	Открытая страница веб приложения
Сценарий аттестационного тестирования	1. Нажать на кнопку «Positive Note» 2. Результат – заметка отобразиться на экране слева 3. Нажать на заметку мышкой за угол и потянуть в сторону 4. Заметка изменилась в размере
Ожидаемый результат	Заметка изменена в размере

№	A4
Тип	Общий
Описание	Создать несколько досок с разным количеством заметок
Начальное состояние	Открытая страница веб приложения
Сценарий аттестационного тестирования	1. Нажать на кнопку «Positive Note» 2. Результат – заметка отобразиться на экране слева 3. Нажать на кнопку Save 4. Результат – рабочее пространство сохранилось в меню справа 5. Нажать на кнопку New – создание нового холста 6. Результат – холст создан 7. Создать несколько разных заметок нажимая на кнопки создания заметок 8. Сохранить Save Текущую рабочую область 9. Закрыть вкладки 10. Открыть вкладки 11. Результат – все рабочие холсты сохранены 12. Открыть любой из них 13. Результат – сохраненные заметки появились на своих позициях на холсте
Ожидаемый результат	Несколько досок создано, каждая со своими заметками, и с сохраненным состоянием заметок

№	A5

Тип	Общий
Описание	Создать несколько заметок и экспортировать их в файл
Начальное состояние	Открытая страница веб приложения
Сценарий аттестационного тестирования	1. Нажать на кнопку «Positive Note» 3 раза 2. Результат – заметки отобразятся на экране слева 3. Написать текст на каждой 4. Нажать на кнопку – Export 5. Выбрать расположение файла и имя
Ожидаемый результат	Заметки сохраняются в txt файл

3.4 Нагрузочное тестирование

№	C1
Тип	Специальный
Начальное состояние	Открытая страница веб приложения
Описание	1.Создать 50 заметок, расположить их хаотично на рабочем холсте. 2.Сохранить изменения 3. Нажать на кнопку в меню «New» - создать новый борд. 4. Создать 50 заметок, расположить их хаотично на рабочем холсте. 5. Выполнить 3 – 4 5 раз 6.Открыть 10 новых вкладок 7. Убедиться, что данные сохранились, сервис работает. 4.Перезапустить браузер
Ожидаемый результат	Все данные сохранены, сервис продолжает работать

3.5 Тестовое покрытие

Сложность современного программного обеспечения и инфраструктуры сделало невыполнимой задачу проведения тестирования со 100% тестовым покрытием. Расчет тестового покрытия относительно исполняемого кода программного обеспечения проводится по формуле:

$$T_{cov} = \frac{L_{tc}}{L_{code}} * 100\%$$, где:

- T_{cov} - тестовое покрытие
- L_{tc} - количество строк кода, покрытых тестами
- L_{code} - общее количество строк кода

Общее количество строк javascript кода vue компонентов приложения – 470 строк.

Покрыто тестами - 173 строки.

Таким образом, тестовое покрытие составляет – 36.8%.

Пример реализации теста

Пример реализации теста Б8 функции createBoard(initial) модуля App:

```
import { mount } from '@vue/test-utils'
import App from '~/App.vue'

describe('createBoard()', () => {
    it('type of argument isn’t boolean - throw error', () => {
        const wrapper = mount(App);
        result = wrapper.vm.createBoard('Hello World!');
        expect(result.message).toBe('The type of arg must be Boolean!');
    })
})
```

4.Журнал тестирования

Блочное

Номера тестов	Объект	Количество тестов	Количество ошибок	Дата	Тестировщик
Б1	data()	1	0	9.12.2018	Лайтинен Н.В
Б2	activeBoard ()	1	0	9.12.2018	Лайтинен Н.В
Б3	createBoard (initial)	1	0	9.12.2018	Лайтинен Н.В
Б4, Б11	loadState()	2	1 ошибка в Б11 Отчет об ошибке №1	9.12.2018	Лайтинен Н.В
Б5, Б10	saveState()	2	1 ошибка в Б10 Отчет об ошибке №2	9.12.2018	Лайтинен Н.В
Б6, Б12, Б13	migrateState ()	3	1 ошибка в Б12 Отчет об ошибке №3	9.12.2018	Лайтинен Н.В
Б7	created ()	1	0	9.12.2018	Лайтинен Н.В
Б8, Б9	createBoard (initial)	2	2 ошибки в Б8 и Б9. Отчеты об ошибках №4 и №5	9.12.2018	Лайтинен Н.В

Б14, Б15, Б16, Б17	startDrag(id)	4	2 ошибки в Б16 и Б17 Отчеты об ошибках №6 и №7	9.12.2018	Лайтинен Н.В
Б18, Б19, Б20	updateNote (id, update)	3	2 ошибки в Б18 и Б19 Отчеты об ошибках №8 и №9	9.12.2018	Лайтинен Н.В
Б21, Б22, Б23	addNote (type)	3	3 ошибки в Б21, Б22, Б23 Отчеты об ошибках №10 №11 №12	9.12.2018	Лайтинен Н.В
Б24	incrFontSize()	2	0	9.12.2018	Лайтинен Н.В
Б25, Б26, Б27, Б28	onPositionMouseMove (d)	4	2 ошибки в Б25 и Б26 Отчеты об ошибках №13 №14	9.12.2018	Лайтинен Н.В
Б29, Б30, Б31	onSizeMouseMove (d)	3	2 ошибки в Б29 и Б30 Отчеты об ошибках №15 №16	9.12.2018	Лайтинен Н.В
Б32, Б33, Б34	getPositionforNew()	3	2 ошибки в Б32 и Б33 Отчеты об ошибках №17 №18	9.12.2018	Лайтинен Н.В

Интеграционное

Номера тестов	Этап интеграции	Количество тестов	Количество ошибок	Дата	Тестировщик
И1	1	1	0	9.12.2018	Лайтинен Н.В
И2	2	1	0	9.12.2018	Лайтинен Н.В
И3	3	1	0	9.12.2018	Лайтинен Н.В
И4	4	1	0	9.12.2018	Лайтинен Н.В
И5	5	1	0	9.12.2018	Лайтинен Н.В
И6	6	1	0	9.12.2018	Лайтинен Н.В

Аттестационное

Номера тестов	Функциональные требования	Количество тестов	Количество ошибок	Дата	Тестировщик
А1	1	10	0	9.12.2018	Лайтинен Н.В
А2	4	10	0	9.12.2018	Лайтинен Н.В
А3	2	10	0	9.12.2018	Лайтинен Н.В
А4	6	10	0	9.12.2018	Лайтинен Н.В
А5	9	10	0	9.12.2018	Лайтинен

					Н.В

Нагрузочное

Тест не пройдет. Отчет №19.

5. Журнал найденных ошибок

Отчет об ошибке №1:

Тест: Б11

Объект тестирования: функция loadState ()

Алгоритм: Установить mock window.localstorage = undefined. Вызвать функцию.

Ожидаемый результат: Выброс исключения ‘The LocalStorage isn’t found’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №2:

Тест: Б10

Объект тестирования: функция saveState ()

Алгоритм: Установить mock window.localstorage = undefined. Вызвать функцию.

Ожидаемый результат: Выброс исключения ‘The LocalStorage isn’t found’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №3:

Тест: Б12

Объект тестирования: функция migrateState ()

Алгоритм: Установить mock window.localstorage = undefined. Вызвать функцию.

Ожидаемый результат: Выброс исключения ‘The LocalStorage isn’t found’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №4:

Тест: Б8

Объект тестирования: createBoard (initial)

Алгоритм: передать в функцию любую строку, либо объект, число.

Ожидаемый результат: Выброс исключения ‘The type of arg must be Boolean!”

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №5:

Тест: Б9

Объект тестирования: createBoard (initial)

Алгоритм: вызвать функцию без параметра.

Ожидаемый результат: Выброс исключения ‘The argument isn’t found’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №6:

Тест: Б16

Объект тестирования: startDrag (id)

Алгоритм: вызвать функцию с параметрами типа строки, массива, объекта.

Ожидаемый результат: Выброс исключения “The type of arg must be integer!”

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №7:

Тест: Б17

Объект тестирования: startDrag (id)

Алгоритм: вызвать функцию без параметра.

Ожидаемый результат: Выброс исключения ‘The argument isn’t found’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №8:

Тест: Б18

Объект тестирования: updateNote (id, update)

Алгоритм: вызвать функцию с параметрами (12,23), (привет, [‘1212’]) ([‘wwg’], 12).

Ожидаемый результат: Выброс исключения “The type of args isn’t correct!”

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №9:

Тест: Б19

Объект тестирования: updateNote (id, update)

Алгоритм: вызвать функцию без параметров

Ожидаемый результат: Выброс исключения ‘The arguments aren’t found’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №10:

Тест: Б21

Объект тестирования: addNote (type)

Алгоритм: вызвать функцию с неправильным параметром: 43, [‘1’], {[1]}

Ожидаемый результат: Выброс исключения “The type of args isn’t correct!”

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №11:

Тест: Б22

Объект тестирования: addNote (type)

Алгоритм: вызвать функцию без параметра

Ожидаемый результат: Выброс исключения ‘The argument isn’t found’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №12:

Тест: Б23

Объект тестирования: addNote (type)

Алгоритм: вызвать с параметрами, отличными от имеющихся типов заметки, например ‘hello’, ‘wgwgwg’. ‘another string’.

Ожидаемый результат: Выброс исключения ‘The type of note isn’t correct!’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №13:

Тест: Б25

Объект тестирования: onPositionMouseMove (d)

Алгоритм: вызвать функцию с параметром, отличном от типа объект, например ‘wgwgw’, 42, [1,1,1,23]

Ожидаемый результат: Выброс исключения “The type of arg must be obj!”

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №14:

Тест: Б26

Объект тестирования: onPositionMouseMove (d)

Алгоритм: вызвать функцию без параметра

Ожидаемый результат: Выброс исключения ‘The argument isn’t found’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №15:

Тест: Б29

Объект тестирования: onSizeMouseMove (d)

Алгоритм: вызвать функцию с параметром, отличном от типа объект, например ‘wgwgw’, 42, [1,1,1,23]

Ожидаемый результат: Выброс исключения “The type of arg must be obj!”

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №16:

Тест: Б30

Объект тестирования: onSizeMouseMove (d)

Алгоритм: вызвать функцию без параметра

Ожидаемый результат: Выброс исключения ‘The argument isn’t found’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №17:

Тест: Б32

Объект тестирования: getPositionforNew()

Алгоритм: вызвать функцию с параметром, отличным от типа integer, например “gfqwg”, [1], {1: ‘1r14r’}

Ожидаемый результат: Выброс исключения “The type of arg must be integer!”

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №18:

Тест: Б33

Объект тестирования: getPositionforNew()

Алгоритм: вызвать функцию без параметра

Ожидаемый результат: Выброс исключения ‘The argument isn’t found’

Фактический результат: выброс исключения не произошел

Воспроизводимость: всегда

Дата проведения: 9.12.2018

Отчет об ошибке №19:

Тест: С1

Объект тестирования: интерфейс

Алгоритм: выполнить действия по сценарию теста

Ожидаемый результат: все заметки сохранены, данные в них сохранены, расположения сохранены

Фактический результат: данные – текс сохранен, количество заметок тоже, но расположение не сохранилось, они выстроились как при создании с нуля.

Воспроизводимость: всегда

Дата проведения: 9.12.2018

6. Результаты

Веб-приложение Retrospectify имеет неудачную архитектуру, в связи с чем описанный набор тестов имеет такой вид.

Так как язык javascript не строго типизированный, то по этой причине основные ошибки – халатное отношение к передаваемым аргументам. В теле функций отсутствует какая либо логика для отлова неправильного типа передаваемых параметров, а также вовсе отсутствия этих параметров.

Также хотелось бы отметить, что отсутствует проверка на возможность использования локального хранилища, что является критическим для этого приложения. Все данные хранятся в localstorage.

Для решения этой проблемы можно либо написать соответствующий функционал либо ограничить возможность пользования приложением лицам, у кого ПО не поддерживает localstorage.