Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Языки программирования (ЯП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:

ПРОГРАМНОЕ СРЕДСТВО «ВЕБ-БРАУЗЕР «NET BAR»»

БГУИР КП 1-40 01 01 027 ПЗ

Студент Шестаков И.М.

Руководитель Болтак С.В.

Минск 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Интернет является неотъемлемой частью нашей жизни. Среднестатистический человек не может обойтись без него хотя бы один день. Просмотр предстоящей погоды, расписания транспорта, фильмов и сериалов, навигация при помощи карт и многое другое стало доступно современному человеку в одном месте.

Для более удобной навигации в таком огромном пространстве как интернет были придуманы веб-браузеры. Браузер, или веб-обозреватель - прикладное программное обеспечение для просмотра страниц, содержания веб-документов, компьютерных файлов и их каталогов; управления веб-приложениями; а также для решения других задач.

Первый веб-браузер появился в 1990 году благодаря Тиму Бернерсу-Ли. Назывался он «WorldWideWeb», позже был переименован в «Nexus». Но первым распространённым браузером с графическим интерфейсом был «NCSA Mosaic». Исходный код этого одного из первых браузеров был открыт, и некоторые другие браузеры (Netscape Navigator и Internet Explorer) взяли его за основу. Этот браузер имел свои недостатки, но почти все они были устранены в браузере Netscape Navigator (некоторые сотрудники компании Netscape были из NCSA и участвовали в разработке Mosaic). Netscape выпустила Netscape Navigator под разные операционные системы (UNIX, Windows, Mac OS) и добилась заметного успеха, в том числе и коммерческого. Это побудило компанию Microsoft выпустить свой браузер Internet Explorer.

Борьба производителей браузеров за доминирование получила название Война Браузеров.

На сегодняшний день самыми популярными браузерами являются: «Google Chrome», «Opera», «Mozila Firefox» и «Microsoft Edge». Если взять статистику распространения браузеров во всём мире, то лидирующую позицию с весомым отрывом занимает браузер «Google Chrome», на втором месте находится браузер «Opera».

Функциональные возможности браузеров постоянно расширяются и улучшаются благодаря конкуренции между их разработчиками и высоким темпам развития и внедрения информационных технологий. Несмотря на то, что браузеры разных изготовителей базируются на разных технологических решениях, большинство современных браузеров придерживается международных стандартов и рекомендаций в области обработки и отображения данных.

Тема разработки и сама разработка веб-браузеров является весьма интересной. Поэтому целью данного курсового проекта является разработка программного средства «Веб-браузер «Net bar»».

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

На сегодняшний день существует огромное количество веб-браузеров. Каждый из них может похвастаться своим уникальным функционалом.

В первую очередь стоит рассмотреть, что предлагает веб-браузер, который используется по умолчанию в Windows 10 – «Microsoft Edge». Внешний вид данного приложения представлен на рисунке 1.1.

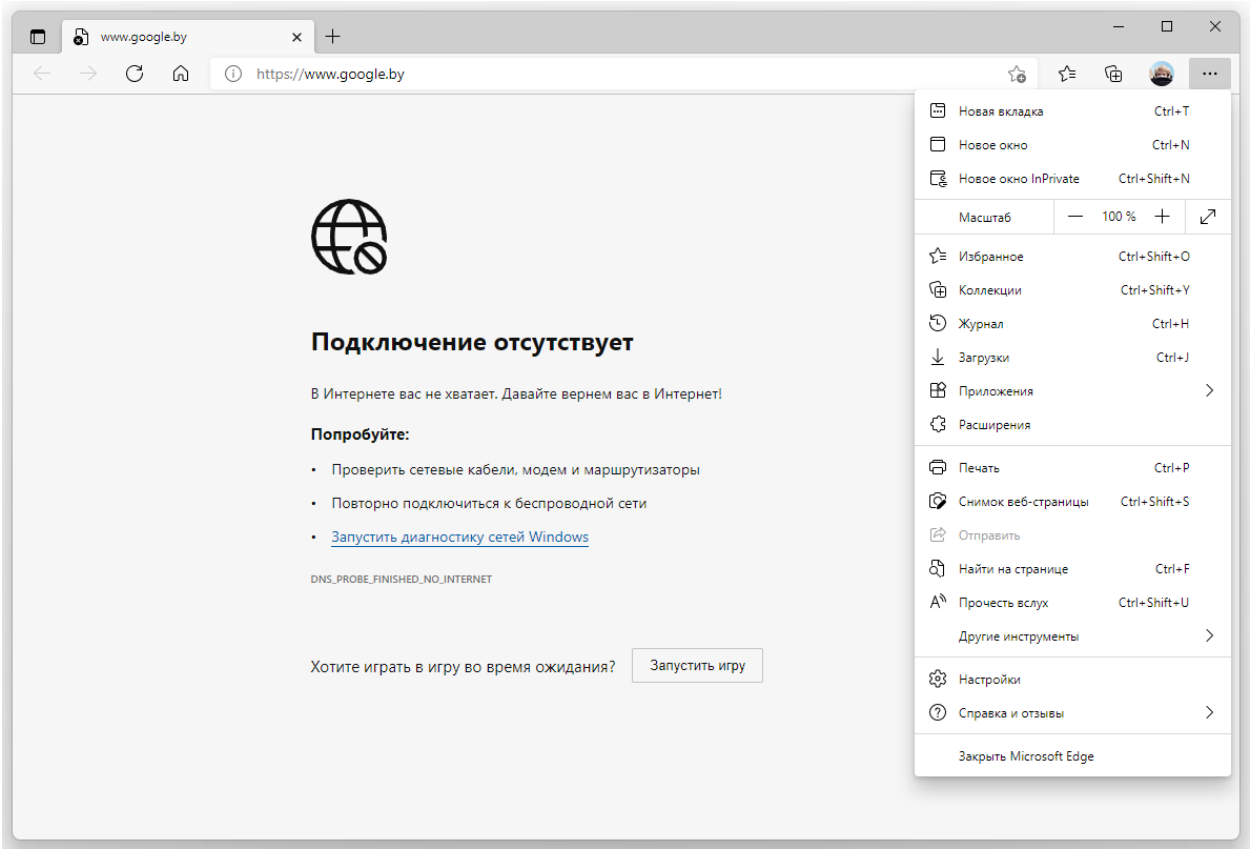

Рисунок 1.1 – браузер «Microsoft Edge»

Данный браузер выполнен по лекалам Chromium. В визуальной стилистике и по части пользовательского интерфейса, а также по расположению элементов управления представляет собой практически точную копию знакомого всем веб-браузера «Google Chrome». Разница кроется в деталях.

Самая важная особенность веб-браузера «Microsoft Edge» - это тесная интеграция с онлайновыми сервисами Microsoft. Браузер оснащён средствами авторизации в Microsoft-аккаунте и синхронизации пользовательских данных на разных устройствах, а также поддержкой поисковой системы «Bing», сервисов информационного портала «Microsoft News», переводчика «Microsoft Translator» и прочих служб софтверного гиганта. При этом

привязка к сервисам Microsoft в большинстве случаев настраиваемая, и от поддержки некоторых служб при желании можно отказаться.

Далее стоит рассмотреть самый популярный на сегодняшний день веб-браузер «Google Chrome». Внешний вид данного приложения представлен на рисунке 1.2.

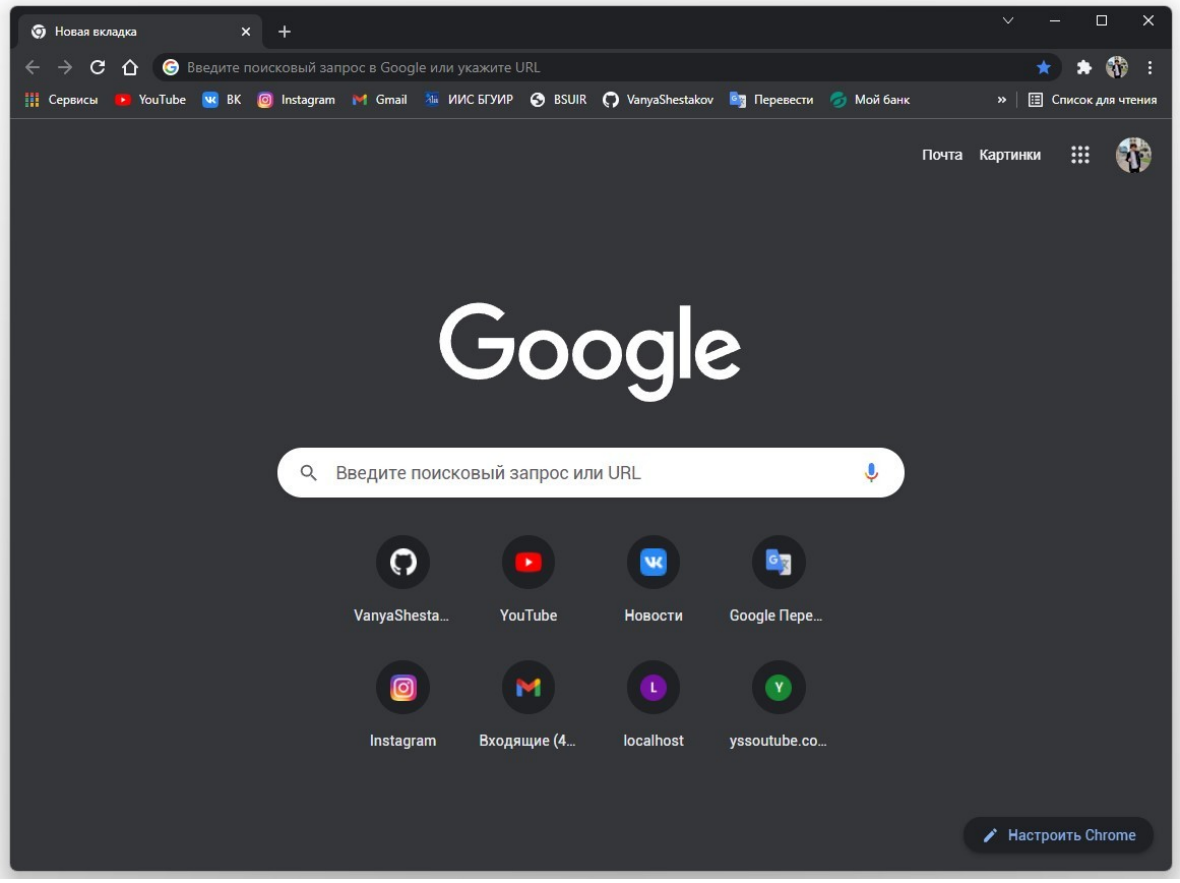

Рисунок 1.2 – браузер «Google Chrome»

Данный браузер не нуждается в представлении. За 12 лет развития он не только заслужил доверие сетевой аудитории и стал самым популярным браузером в мире, но и сумел определить новые стандарты качества, производительности, надёжности и безопасности в своём сегменте.

«Google Chrome» позволяет хранить данные каждого пользователя отдельно, будь то закладки, история просмотров, пароли и настройки браузера. Также браузер поддерживает работу в гостевом режиме. Важной особенностью профилей в «Google Chrome» является то, что их нельзя защитить паролем, а переключаться между ними может любой пользователь. Данный веб-браузер имеет возможность воспроизведения аудио- и видеофайлов без сторонних программ и расширений. Также приятными бонусами браузера «Google Chrome» можно считать наличие встроенного диспетчера задач и антивируса. При взаимодействии с большим количеством вкладок браузер позволяет их группировать. Также стоить отметить такую особенность, как быстрая пересылка данных с браузера на свой смартфон.

1.2 Постановка задачи

В рамках данной курсовой работы планируется разработать веб-браузер для операционной системы Windows. В процессе разработки должны быть реализованы базовые функции веб-браузера:

- навигация по сайтам (вперед, назад, возврат на домашнюю страницу);
- обновление текущего сайта;
- поиск в адресной строке;

Также будут реализованы основные возможности работы со вкладками:

- создание новой вкладки;
- закрытие вкладки;
- переключение между вкладками;

Будут реализованы базовые возможности для работы с закладками:

- добавление новой закладки;
- удаление закладки;
- очистка всех закладок;
- просмотр всех закладок;

Планируется реализовать функционал, связанный с историей посещения сайтов веб-браузера:

- просмотр истории;
- навигация по истории;
- очистка истории;

Также в приложении будут реализованы дополнительные функции:

- «Анонимный режим»;
- возможность очистки кэша браузера;
- возможность очистки всех данных браузера;
- смена домашней страницы.

Для разработки программного средства будет использоваться язык программирования C++ и среда разработки Embarcadero C++ Builder 10.4.

2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

2.1 Структура программы

При разработке приложения будет использовано 14 модулей:

– MainView – модуль, для отображения главного окна приложения;

– BrowserManger – модуль, обеспечивающий работу с данными веб-браузера

– BookmarksManager – модуль, обеспечивающий работу с закладками браузера

– HistoryManager – модуль, обеспечивающий работу с историей посещения веб-сайтов;

– BookmarksReader – модуль, обеспечивающий чтение закладок браузера из файла;

– BookmarksWriter – модуль, обеспечивающий запись закладок браузера в файл;

– HistoryReader – модуль, обеспечивающий чтение истории браузера из файла;

– HistoryWriter – модуль, обеспечивающий запись истории браузера в файл;

– SettingsView – модуль, для отображения окна настроек браузера;

– AddBookmarkView – модуль, для отображения окна добавления закладки;

– AnonymModeView – модуль, для отображения информации об «Анонимном режиме»;

– AboutProgramView – модуль, для отображения информации о приложении;

– SiteVisit – модуль, хранящий информацию о посещенном сайте;

– StringConverter – модуль, обеспечивающий конвертацию строк;

2.2 Проектирование интерфейса программного средства

При разработке программного средства за основу будет взят стандартный внешний вид веб-браузера.

2.2.1 Главное окно

Главное окно приложения должно состоять из двух основных частей. В верхней части окна будет находится адресная строка и элементы управления браузером, ниже элементов управления будет находится область для отображения веб-контента. Макет главного окна веб-браузера представлен на рисунке 2.1.

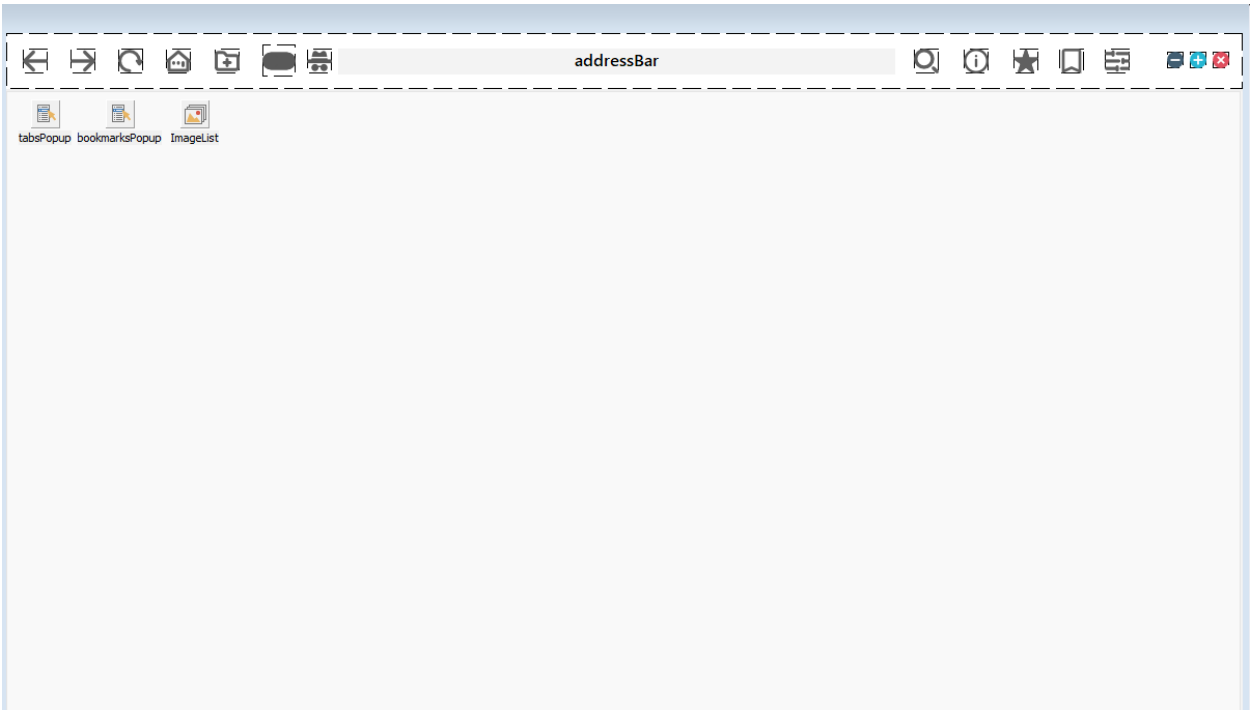

Рисунок 2.1 – Главное окно приложения

2.2.2 Элементы управления веб-браузером

Элементы управления браузером должны включать в себя следующие функциональные компоненты:

a) Клавиши (за основу будет взят компонент TImage) для обеспечения функционала:
   1) Возврат на предыдущую страницу;
   2) Переключение на следующую страницу;
   3) Обновление текущей страницы;
   4) Возврат на домашнюю страницу;
   5) Открытие новой вкладки;
   6) Включение анонимного режима;
   7) Поиск запроса в интернете;
   8) Просмотр информации о приложении;
   9) Добавление/удаление закладки;
   10) Просмотр списка закладок;
   11) Открытие окна настроек;
b) Адресная строка (компонент TEdit). Для просмотра адреса текущего сайта или поиска запроса в интернете;
c) Список закладок (компонент TComboBox). Для просмотра закладок;
d) Вкладки (компонент TPageControl);

При наведении на каждый элемент управления появляется всплывающая подсказка, говорящая о предназначении компонента.

Внешний вид элементов управления веб-браузером представлен на рисунке 2.2.

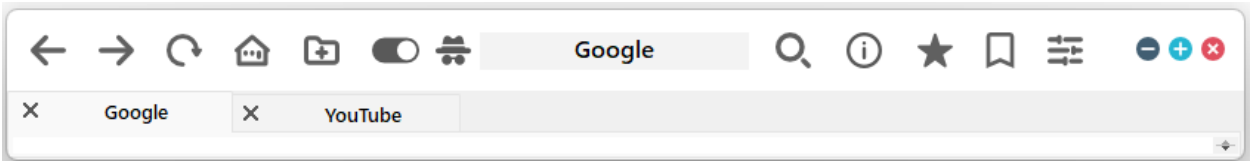

Рисунок 2.2 – Элементы управления веб-браузером

2.2.3 Окно настроек программного средства

Окно настроек будет состоять из двух частей. В левой части окна будут находится кнопки (компонент TBitBtn) для выполения различных задач (очистка кэша, очистка данных браузера, очистка закладок, просмотр истории), также будет возможность смены домашней страницы при помощи компонентов TEdit (для указания адреса новой домашней страницы) и TBitBtn (для подтверждения смены домашней страницы).

Правая часть окна будет разворачиваться по нажатию кнопки в левой части окна. В правой части окна будет находится история посещения сайтов (компонент TListBox) и кнопка для очистки истории (TBitBtn). По списку истории можно осуществлять навигацию. Макет окна настроек представлен на рисунке 2.3

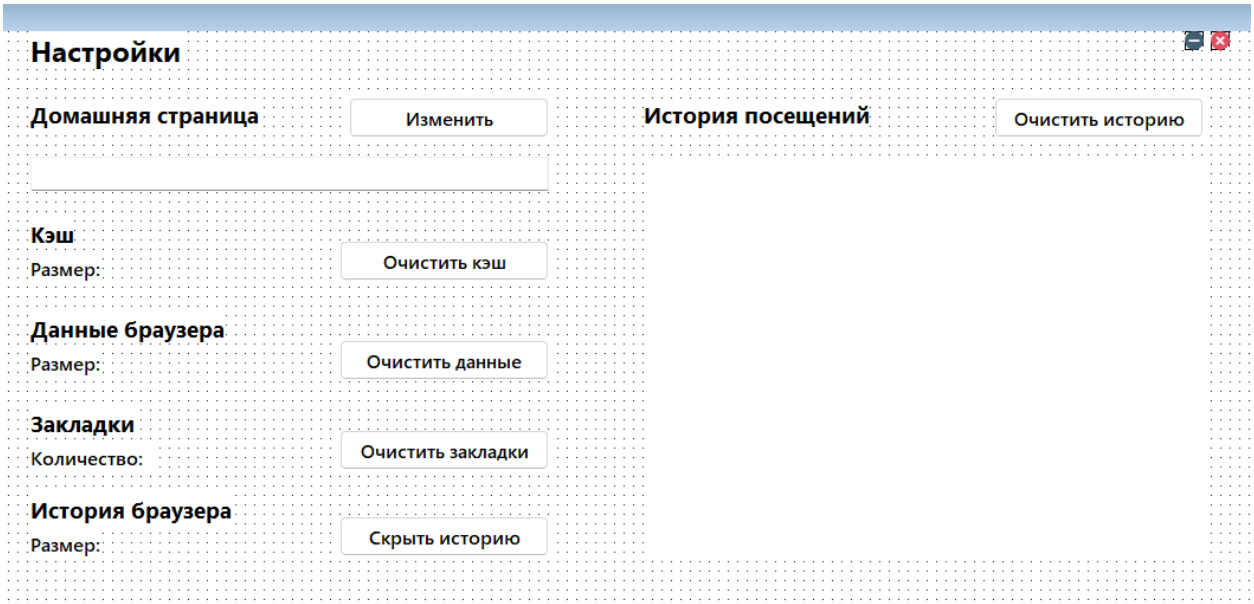

Рисунок 2.3 – Макет окна настроек веб-браузера

2.2.4 Информационные окна

Программное средство будет содержать два информационных окна: первое – для отображения информации об анонимном режиме, второе – для отображения информации о программном средстве. Внешний вид информационных окон представлен на рисунке 2.4.

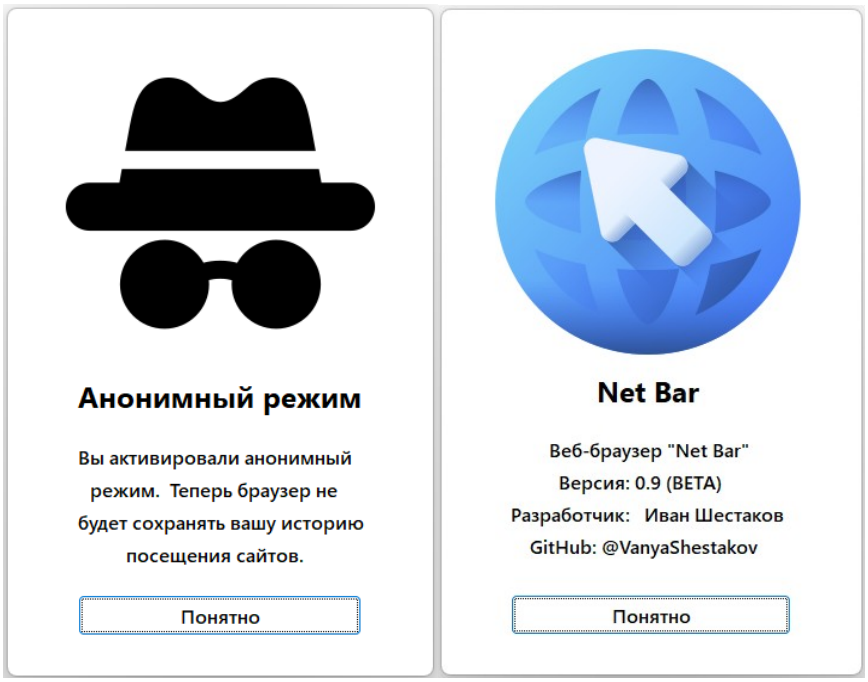

Рисунок 2.4 – Внешний вид информационных окон

2.2.5 Окно добавления закладки

Окно добавления закладки будет содержать две кнопки для добавления/отмены добавления закладки (компонент TBitBtn) и поле ввода названия закладки (компонент TEdit). Внешний вид макета окна добавления закладок представлен на рисунке 2.5.

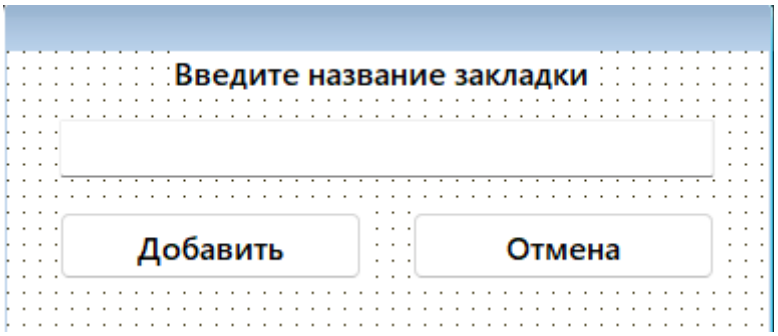

Рисунок 2.5 – Внешний макета окна добавления закладок

2.3 Проектирование функционала программного средства

Граммотно поставленная задача и хорошо составленные алгоритмы – ключевая фаза в проектировании программного средства. Веб-браузер должен предоставлять пользователю такой минимальный функционал как:

- поиск запроса в интернете;
- добавление закладки;
- удаление закладки.

2.3.1 Поиск запроса в интернете

Поиск запроса в интернете будет происходит по нажатию кнопки «Найти» или по нажатию клавиши «Enter» на клавиатуре. Сам запрос будет прописываться в адресной строке. Если запрос будет являться ссылкой на сайт, то браузер автоматически перейдет по данной ссылке, иначе будет выполнен поисковой запрос. Блок-схема кода, осуществляющего поиск информации в интернете, представлена на рисунке 2.6.

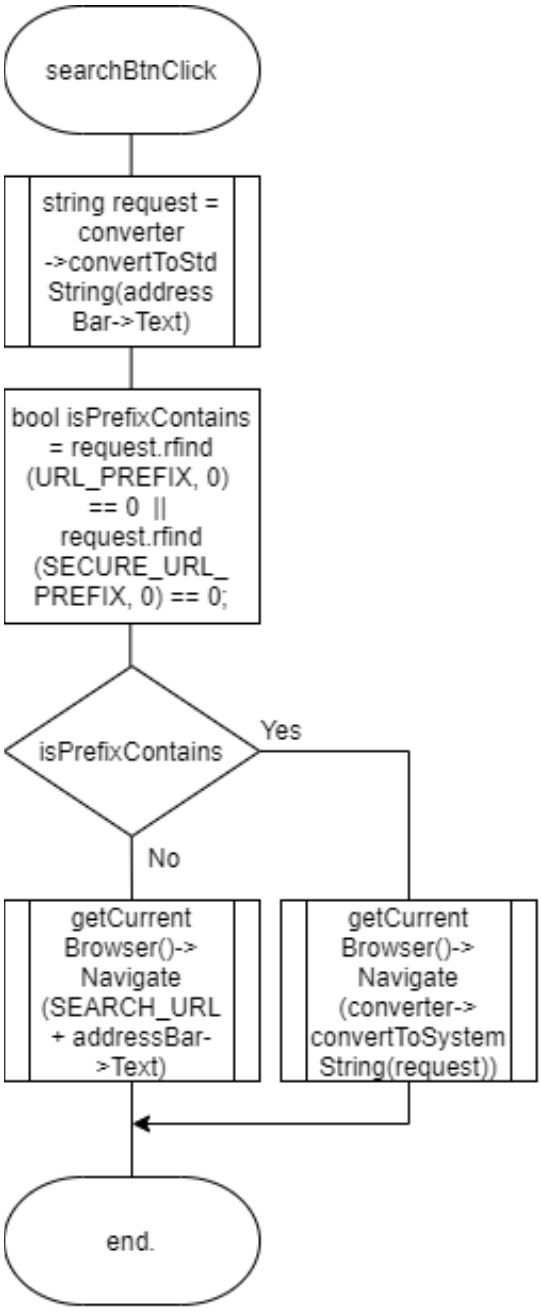

Рисунок 2.6 – Блок-схема функции searchBtnClick()

2.3.2 Добавление закладки

Добавление закладки будет происходит по нажатию кнопки «Добавить страницу в закладки». После нажатия данной кнопки должно будет открыться окно, в котором пользователь сможет задать имя новой закладки и добавить ее. При добавлении закладки будет вызываться метод addBookmark() класса BookmarksManager. В качестве параметров он будет принимать заголовок страницы и адрес. После добавления в список закладок, закладка запишется в файл. Блок-схема метода addBookmark() представлена на рисунке 2.7.

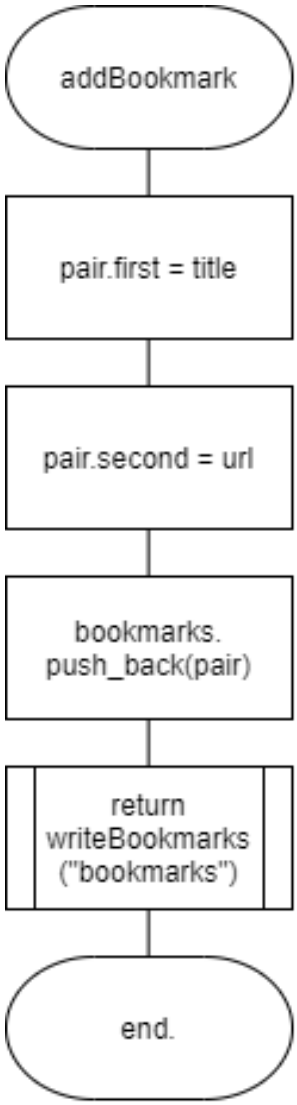

Рисунок 2.7 – Блок-схема метода addBookmark()

2.3.3 Удаление закладки

Удаление закладки будет происходит по нажатию кнопки «Удалить страницу из закладок». При удалении закладки будет вызываться метод removeBookmark() класса BookmarksManager. В качестве параметров он будет принимать адрес страницы. Далее циклом будет происходить поиск нужного адреса сайта среди списка закладок. После нахождения нужного адреса в списке закладок, запоминается индекс данной закладки. Далее по данному

индексу будет произведено удаление нужной закладки. Блок-схема метода removeBookmark() представлена на рисунке 2.8.

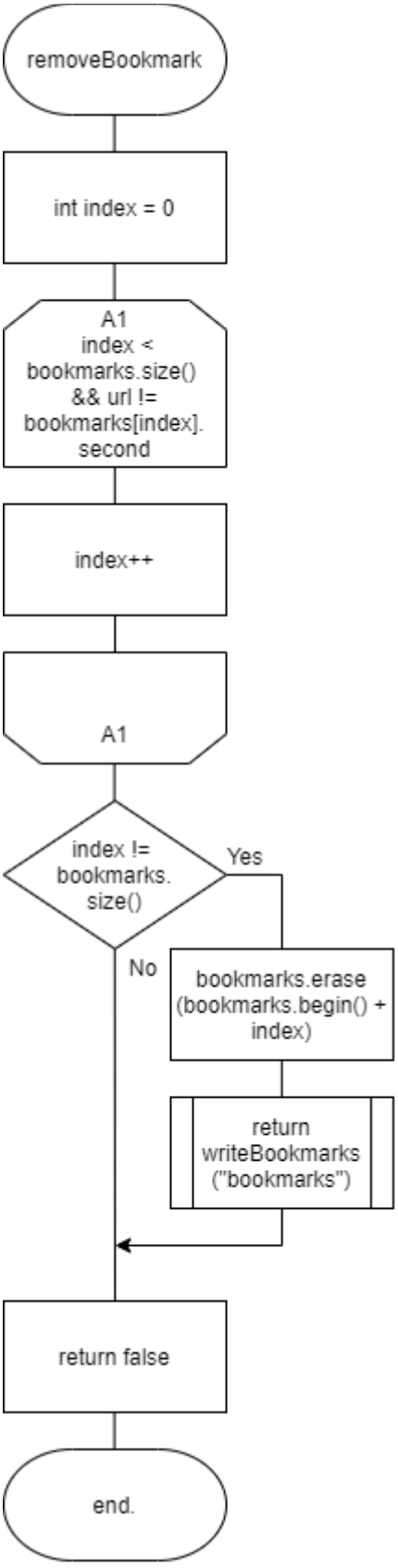

Рисунок 2.8 – Блок-схема метода removeBookmark()

3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

3.1 Навигация по сайтам

Ключевыми возможностями программного средства являются функции, связанные с навигацией по сайтам. Навигация включает себя переход по страницам, возврат на предыдущую страницу, переключение на следующую страницу, обновление текущей страницы и возврат на домашнюю страницу. В основу работы приложения был положен компонент TWebBrowser, который обеспечивает простую навигацию внутри браузера.

3.1.1 Функции «Вперед», «Назад», «Обновить», «Вернуться домой»

Навигация внутри браузера осуществляется по нажатию кнопок. В обработчиках событий каждой кнопки вызываются соответствующие методы компонента TWebBrowser: метод GoBack() для возврата назад, метод GoForward() для переключения на следующую страницу, метод Refresh() для обновления текущей страницы и метод Navigate для возврата на домашнюю страницу. Код обработчиков событий функций связанных с навигацией представлен ниже.

```
void __fastcall TWebView::backBtnClick(TObject *Sender)
{
    getCurrentBrowser()->GoBack();
}

void __fastcall TWebView::forwardBtnClick(TObject *Sender)
{
    getCurrentBrowser()->GoForward();
}

void __fastcall TWebView::updateBtnClick(TObject *Sender)
{
    getCurrentBrowser()->Refresh();
}

void __fastcall TWebView::homeBtnClick(TObject *Sender)
{
    getCurrentBrowser()->Navigate(homepageUrl);
}
```

Из представленного кода можно увидеть, что метод Navigate() принимает в качестве параметра путь к домашней странице браузера, так как данный путь может изменяться программно.

3.1.2 Реализация поиска в адресной строке

Поиск осуществляется по нажатию кнопки «Найти» или по нажатию клавиши «Enter». Обработчики событий вызывают метод search(), который на

вход принимает строковое представление запроса поиска, сам запрос берется из адресной строки веб-браузера. Код метода search() представлен ниже.

```
void TWebView::search(std::string request)
{
    bool isPrefixContains = request.rfind(URL_PREFIX, 0) == 0  ||
                            request.rfind(SECURE_URL_PREFIX, 0) == 0;
    if (isPrefixContains)
    {
        getCurrentBrowser()->Navigate(converter->convertToSystemString(request));
    }
    else
    {
        getCurrentBrowser()->Navigate(SEARCH_URL + addressBar->Text);
    }
}
```

Данный метод изначально проверяет содержание префикса «https://» или «http://» в запросе. Если такой префикс содержится, то значит, что пользователь хочет перейти непосредственно на искомую страницу и браузер выполняет переход. Если же запрос не содержит данных префиксов, то будет происходить поиск запроса в поисковой системе «Google».

3.2 Реализация вкладок

Очень часто современному пользователю хочется держать во внимании большое количество информации в интернете, запускать для этого несколько окон приложения браузер совсем неудобно. Решением данной проблемы будет создания независимых вкладок.

3.2.1 Создание вкладки

Создание вкладки происходит по нажатию кнопки «Открыть новую вкладку». Обработчик события данной кнопки вызывает метод createNewTab(). Код данного метода приведен ниже.

```
void TWebView::createNewTab()
{
    TTabSheet *tab = new TTabSheet(PageControl);
    tab->Caption = NEW_TAB_CAPTION;
    tab->PageControl = PageControl;
    PageControl->ActivePage = tab;
    tab->Name = TAB_NAME_PREFIX + String(tabId);

    TWebBrowser *browser = new TWebBrowser(PageControl);
    browser->SelectedEngine = TWebBrowser::TSelectedEngine::EdgeOnly;
    browser->TOleControl::Parent = tab;
    browser->Width = tab->Width;
    browser->Height = tab->Height;
    browser->Align = alClient;
    browser->Navigate(homepageUrl);
    tab->Tag = Integer(browser);
    browser->TOleControl::Name = tab->Name + BROWSER_NAME_POSTFIX;
```

```
    browser->OnDocumentComplete = DocumentComplete;
    browser->OnNewWindow3 = NewWindow3;
    browser->OnTitleChange = TitleChange;
    browser->OnBeforeNavigate2 = BeforeNavigate2;
    tabId++;
}
```

Как видно из данного метода, для работы со вкладками был использован компонент TPageControl. В первую очередь динамически создается сама вкладка, далее она привязывается к компоненту TPageControl, ей задается имя. После связывания созданной вкладки с родительским компонентом динамически создается компонент TWebBrowser для данной вкладки. Далее происходит настройка свойств компонента TWebBrowser, после настройки ему назначаются реализации событий OnDocumentComplete, OnNewWindow3, OnTitleChange и OnBeforeNavigate2.

3.2.2 Закрытие вкладки

Закрытие вкладки происходит по нажатию кнопки «Крестик» около названия вкладки. Для отображения данной кнопки был использован компонент TImageList, который был привязан к компоненту TPageControl. Закрытие вкладки происходит по совершению события OnMouseUp компонента TPageControl. Код обработчика данного события представлен ниже.

```
void __fastcall TWebView::PageControlMouseUp(TObject *Sender, TMouseButton Button,
          TShiftState Shift, int X, int Y)
{
    const int CLOSE_BTN_SIZE = 20;
    if (Button == mbLeft)
    {
        TRect tabRect = PageControl->TabRect(PageControl->ActivePage->TabIndex);

        bool isClose = PtInRect(
            Classes::Rect(tabRect.Left, tabRect.Top,
            tabRect.Left + CLOSE_BTN_SIZE, tabRect.Top + CLOSE_BTN_SIZE),
            Classes::Point(X, Y));

        if (isClose && PageControl->PageCount > 1)
        {
            closeTab(PageControl->IndexOfTabAt(X, Y));
        }
    }
}
```

В данном обработчике события отслеживается в каком месте на вкладке была нажата кнопка «Крестик». Для этого создается объект класса TRect, который представляет из себя фигуру «прямоугольник». Данный прямоугольник создается на основе размеров вкладки. Далее при помощи метода PtInRect() происходит проверка содержания точки нажатия мыши в новом прямоугольнике, который создан на основе первого но со смещенными координатами, которые задают размер кнопки «Крестик». Если же точка нажатия мыши находится в данном прямоугольнике, значит пользователь

нажал на кнопку «Крестик», следовательно следует закрыть данную вкладку. Для закрытия данной вкладки происходит вызов метода closeTab(), который принимает индекс закрываемой вкладки в качестве параметра. Код метода closeTab() приведен ниже.

```
void TWebView::closeTab(int index)
{
    PageControl->Page[index]->Free();
    setLoadingStatus(false);
    if (index != PageControl->PageCount)
    {
        PageControl->ActivePageIndex = index;
    }
    else
    {
        PageControl->ActivePageIndex = index - 1;
    }
    title = getCurrentBrowser()->LocationName;
    pageURL = getCurrentBrowser()->LocationURL;
    addressBar->Text = title;
    setAddBookmarkBtnVisible(!bookmarksManager->contains(converter
                                ->convertToStdString(pageURL)));

}
```

В первую очередь происходит очистка памяти, выделенной под вкладку и под все компоненты, находящиеся на ней. Далее, сбрасывается статус загрузки страницы, чтобы избежать неверного отображения индикатора загрузки в случае, если страница на удаляемой вкладке еще не успела загрузится. В самом конце происходит установка другой активной вкладки.

3.2.3 Переключение вкладок

Во время переключения вкладок браузера срабатывает событие OnChange компонента TPageControl. Код обработчика данного события представлен ниже.

```
void __fastcall TWebView::PageControlChange(TObject *Sender)
{

    title = getCurrentBrowser()->LocationName;
    pageURL = getCurrentBrowser()->LocationURL;
    addressBar->Text = title;
    setAddBookmarkBtnVisible(!bookmarksManager->contains(converter
                              ->convertToStdString(pageURL)));
}
```

В данном методе происходит получение заголовка и адреса страницы с вкладки, на которую произошло переключение. Для получения этих данных нужно определить с какого именно компонента TWebBrowser приходит информация. Для определения компонента TWebBrowser, который находится на текущей вкладке вызывается метод getCurrentBrowser(). Код данного метода представлен ниже.

```
TWebBrowser* TWebView::getCurrentBrowser()
{
    return (TWebBrowser*) PageControl
        ->ActivePage
        ->FindChildControl(PageControl->ActivePage->Name + BROWSER_NAME_POSTFIX);
}
```

Данный метод возвращает компонент TWebBrowser, который извлекается из активной вкладки как ее потомок. Извлечение происходит по имени компонента TWebBrowser.

3.3 Работа с закладками веб-браузера

Для работы с закладками был создан отдельный модуль BookmarksManager, который позволяет выполнять базовые операции над закладками: добавление, удаление и очистка.

3.3.1 Добавление закладки

Добавление страницы в закладки происходит по нажатию кнопки «Добавить страницу в закладки». После этого открывается окно, в котором пользователь может ввести имя закладки или использовать стандартное. После подтверждения добавления закладки происходит вызов метода addBookmark() класса BookmarksManager. Код данного метода приведен ниже.

```
bool BookmarksManager::addBookmark(std::string title, std::string url)
{
    std::pair<std::string, std::string> pair;
    pair.first = title;
    pair.second = url;
    bookmarks.push_back(pair);
    return writeBookmarks("bookmarks");
}
```

Закладки хранятся в динамическом массиве std::vector, для хранения закладки был использован класс std::pair, где первым значением хранился заголовок страницы, а вторым соответствующий адрес. В конце данного метода происходит запись закладок в файл, для их сохранения после закрытия веб-браузера. Для чтения и записи закладок были созданы отдельные модули BookmarksReader и BookmarksWriter соответственно. Метод addBookmark() возвращает булевое значение «true», если запись в файл произошла успешно, и значение «false», если запись произошла неудачно.

3.3.2 Удаление закладки

Удаление страницы из закладок происходит по нажатию кнопки «Удалить страницу из закладок». После нажатия кнопки происходит вызов метода removeBookmark() класса BookmarksManager. Код данного метода приведен ниже.

```
bool BookmarksManager::removeBookmark(std::string url)
{
    int index = 0;
    while (index < bookmarks.size() && url != bookmarks[index].second)
    {
        index++;
    }
    if (index != bookmarks.size())
    {
        bookmarks.erase(bookmarks.begin() + index);
        return writeBookmarks("bookmarks");
    }
    return false;
}
```

Данный метод принимает на вход адрес удаляемой закладки. Далее при помощи цикла происходит поиск нужной закладки, соответствующей данному адресу. После нахождения удаляемой закладки происходит ее удаление при помощи метода erase() класса std::vector из внутреннего динамического массива. В конце данного метода происходит перезапись закладок в файл, для сохранения изменений. Метод removeBookmark() возвращает булевое значение «true», если перезапись в файл произошла успешно, и значение «false», если перезапись произошла неудачно или удаляемая закладка не была найдена.

3.3.3 Открытие закладки

Открытие закладки происходит по выбору нужной закладки из выпадающего списка (TComboBox). При выборе нужной закладки срабатывает событие OnSelect компонента TComboBox. Код обработчика данного события приведен ниже.

```
void __fastcall TWebView::bookmarksBoxSelect(TObject *Sender)
{
    auto bookmarks = bookmarksManager->getBookmarks();
    String url = converter->convertToSystemString(bookmarks[bookmarksBox
                                                  ->ItemIndex].second);
    getCurrentBrowser()->Navigate(url);
}
```

В начале метода происходит получение списка закладок из класса BookmarksManager. Далее по известному индексу выбранной закладки происходит получение адреса нужной страницы. В конце при помощи метода Navigate() осуществляется переход по найденному адресу.

3.4 Работа с историей посещения сайтов веб-браузера

Для работы с историей посещения сайтов был создан отдельный модуль HistoryManager, который позволяет сохранять историю браузера. Также был создан отдельный модуль SiteVisit, который позволяет хранить информацию о

посещении сайта: заголовок сайта, адрес и время посещения. Заголовочный файл класса SiteVisit представлен ниже.

```
#include <string>

class SiteVisit
{
    private:
        std::string title;
        std::string url;
        std::string time;
    public:
        SiteVisit(std::string title, std::string url, std::string time);
        std::string getTitle();
        std::string getUrl();
        std::string getTime();
};
```

3.4.1 Регистрация посещения веб-сайтов

При каждом переходе по страницам веб-браузера и последующих их загрузках происходит событие OnDocumentComplete компонента TWebBrowser. В обработчике данного события происходит вызов метода registerSiteVisit(). Код данного метода приведен ниже.

```
void TWebView::registerSiteVisit(std::string title, std::string url)
{
    time_t now = time(0);
    std::string stringTime = ctime(&now);
    historyManager->addSiteVisit(stringTime, title, url);
}
```

Данный метод регистрирует посещение сайта. Создается объект структуры time_t, который хранит в себе текущее время. Далее данный объект конвертируется в строковое представление. В конце происходит вызов метода addSiteVisit() класса HistoryManager, который и сохраняет регистрацию посещения сайта. Код метода addSiteVisit() представлен ниже.

```
bool HistoryManager::addSiteVisit(std::string time, std::string title, std::string
url)
{
    SiteVisit *visit = new SiteVisit(title, url, time);
    history.push_back(*visit);
    return writeHistory("history");
}
```

Перед сохранением информации о посещении сайта создается объект класса SiteVisit, которому передается в конструктор нужная информация о регистрации посещения веб-страницы. Для хранения истории был выбран динамический массив std::vector, который хранит в себе объекты класса SiteVisit. После добавления информации о посещении сайта в массив происходит перезапись файла, который хранит историю. Для чтения и записи

истории посещения сайтов созданы отдельные модули HistoryReader и HistoryWriter соответственно. Метод addSiteVisit() возвращает булевое значение «true», если запись в файл произошла успешно, и значение «false», если запись произошла неудачно.

3.4.2 Навигация по истории веб-браузера

Просмотреть историю веб-браузера можно в настройках приложения. Для вывода списка истории был использован компонент TListBox. По двойному нажатию на нужный веб-сайт срабатывает событие OnDblClick. В обработчике данного события описан следующий код:

```
void __fastcall TSettingsForm::historyBoxDblClick(TObject *Sender)
{
    int size = WebView->historyManager->getSize();
    int index = size - historyBox->ItemIndex - 1;
    std::vector<SiteVisit> history = WebView->historyManager->getHistory();
    WebView->createNewTab();
    WebView->getCurrentBrowser()->Navigate(history[index].getUrl().c_str());
    SettingsForm->Close();
}
```

В первую очередь данный метод определяет индекс выбранного веб-сайта. Далее идет получение списка истории при помощи объекта класса HistoryManager. После этого создается новая вкладка для открытия веб-сайта. В конце происходит навигация при помощи метода Navigate() компонента TWebBrowser. В качестве параметра метод Navigate() получает адрес веб-сайта из списка истории по нужному индексу.

3.5 Работа с данными веб-браузера

Работа с данными, которые собирает веб-браузер, осуществляется в настройках приложения. В основном это очистка данных, таких как кэш, пароли, закладки и история веб-браузера. Для работы с данными браузера был создан отдельный модуль BrowserManager.

В окне настроек для каждых данных выводится информация о занимаемой ими памяти в мегабайтах. Получить эту информацию можно получить при помощи методов getCasheSize() для кэша и getBrowserDataSize() для всех данных браузера. Данные методы вызывают метод getFolderSize(), который принимает в качестве параметров путь анализируемой директории и переменную, которая будет хранить в себе размер директории. Код метода getFolderSize() представлен ниже.

```
int BrowserManager::getFolderSize(AnsiString folder, int &size)
{
    TSearchRec searchRec;
    if (folder[folder.Length()] == '\\')
    {
        folder.SetLength(folder.Length() - 1);
```

```
    }

    if (FindFirst(folder + "\\*.*", faAnyFile, searchRec) == 0)
    {
        do
        {
            if (searchRec.Name != "." && searchRec.Name != "..")
            {
                if ((searchRec.Attr & faDirectory) != 0)
                {
                    getFolderSize(folder + "\\" + searchRec.Name, size);
                }
                else
                {
                    size += searchRec.Size;
                }
            }
        } while (FindNext(searchRec) == 0);
    }
    FindClose(searchRec);
    return size;
}
```

Данный метод использует объект класса TSearchRec для подсчета количества занимаемой памяти. Метод является рекурсивным, так как любая директория представляет из себя структуру «дерево», соответственно для полного анализа директории необходимо рекурсивно обойти все ее поддиректории.

3.5.1 Очистка кэша веб-браузера

Очистка кэша веб-браузера происходит по нажатию кнопки «Очистить кэш». Обработчик события данной кнопки вызывает метод clearCashe() класса BrowserManager. Код данного метода представлен ниже.

```
void BrowserManager::clearCashe()
{
    std::filesystem::remove_all(CASHE_PATH.c_str());
}
```

Для удаления директории, которая содержит в себе кэш программы был использован метод remove_all() который находится внутри пространства имен std::filesystem. Данный метод принимает в качестве параметра путь удаляемой директории. После очистки кэша приложение должно будет перезагрузится для последующей корректной работы.

3.5.2 Очистка всех данных веб-браузера

Очистка всех данных включает в себя удаление всех паролей, файлов cookie и кэша веб-браузера. Очистка данных веб-браузера происходит по нажатию кнопки «Очистить данные». Обработчик события данной кнопки вызывает метод clearBrowserData() класса BrowserManager. Предварительно у

пользователя спрашивается подтверждение очистки всех данных браузера. Код метода clearBrowserData() представлен ниже.

```
void BrowserManager::clearBrowserData()
{
    std::filesystem::remove_all(BROWSER_DATA_PATH.c_str());
}
```

Для удаления директории, которая содержит в себе данные программы был использован метод remove_all() который находится внутри пространства имен std::filesystem. После очистки данных приложение должно будет перезагрузится для последующей корректной работы.

3.5.3 Очистка закладок веб-браузера

Очистка закладок веб-браузера происходит по нажатию кнопки «Очистить закладки». Обработчик события данной кнопки вызывает метод clearBookmarks() класса BookmarksManager. Код данного метода приведен ниже.

```
bool BookmarksManager::clearBookmarks()
{
    bookmarks.clear();
    return writeBookmarks("bookmarks");
}
```

В первую очередь происходит очистка массива, который хранит в себе закладки веб-браузера. Далее вызовом метода writeBookmarks() происходит перезапись файла с закладками. Данный метод возвращает булевое значение «true», если перезапись в файл произошла успешно, и значение «false», если перезапись произошла неудачно.

3.5.4 Очистка истории посещения веб-сайтов браузера

Очистка истории веб-браузера происходит по нажатию кнопки «Очистить историю». Обработчик события данной кнопки вызывает метод clearHistory() класса HistoryManager. Код данного метода приведен ниже.

```
bool HistoryManager::clearHistory()
{
    history.clear();
    return writeHistory("history");
}
```

В первую очередь происходит очистка массива, который хранит в себе историю веб-браузера. Далее вызовом метода writeHistory() происходит перезапись файла с историей браузера. Данный метод возвращает булевое значение «true», если перезапись в файл произошла успешно, и значение «false», если перезапись произошла неудачно.

4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В ходе тестирования приложения были выявлены некоторые ошибки и недочеты в работе программного средства.

Основной недочет – это «бесконечная загрузка веб-страницы». Проблема была связана с неверным отображением компонента TActivityIndicator, который сигнализирует о загрузке страницы. Возникала она в случае открытия новой вкладки и, не дожидаясь окончательной загрузки страницы, последующего закрытия данной вкладки. Индикатор не останавливался и продолжал свою работу. Проблема была решена путем принудительной остановки работы индикатора при закрытии вкладки веб-браузера. Для этого в метод closeTab(), который отвечает за закрытие вкладки, был добавлен вызов метода setLoadingStatus(), который на вход в качестве параметра принимает булевое значение «false», что сигнализирует об остановке работы индикатора TActivityIndicator. Код метода setLoadingStatus() представлен ниже.

```
void TWebView::setLoadingStatus(bool status)
{
    isLoaded = !status;
    if (status)
    {
        activityIndicator->StartAnimation();
    }
    else
    {
        activityIndicator->StopAnimation();
    }
    updateBtn->Visible = !status;
    activityIndicator->Visible = status;
}
```

Данный метод в зависимости от значения параметра устанавливает статус загрузки страницы посредством изменения значения поля isLoaded класса TWebView. Далее в зависимости от значения параметра устанавливается состояние компонента TActivityIndicator: «выполнить анимацию загрузки» или «остановить анимацию загрузки». В конце метода устанавливается видимость кнопки «Обновить страницу», если значение параметра функции принимает значение «true», то кнопка скрыта, если значение параметра «false», то кнопка видна пользователю. Это происходит в силу того, что компонент TActivityIndicator находится на том же месте, что и кнопка «Обновить страницу». В случае загрузки страницы кнопка «Обновить страницу» исчезает и на ее месте появляется индикация загрузки.

Большинство проблем возникло из-за недочетов на стадии проектирования программного средства, на стадии тестирования приложения все проблемы были исправлены.

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

5.1 Интерфейс программного средства

5.1.1 Главное окно

Главное окно приложения состоит из двух основных частей. В верхней части окна будет находится адресная строка и элементы управления браузером, ниже элементов управления будет находится область для отображения веб-контента. Внешний вид главного окна веб-браузера представлен на рисунке 5.1.

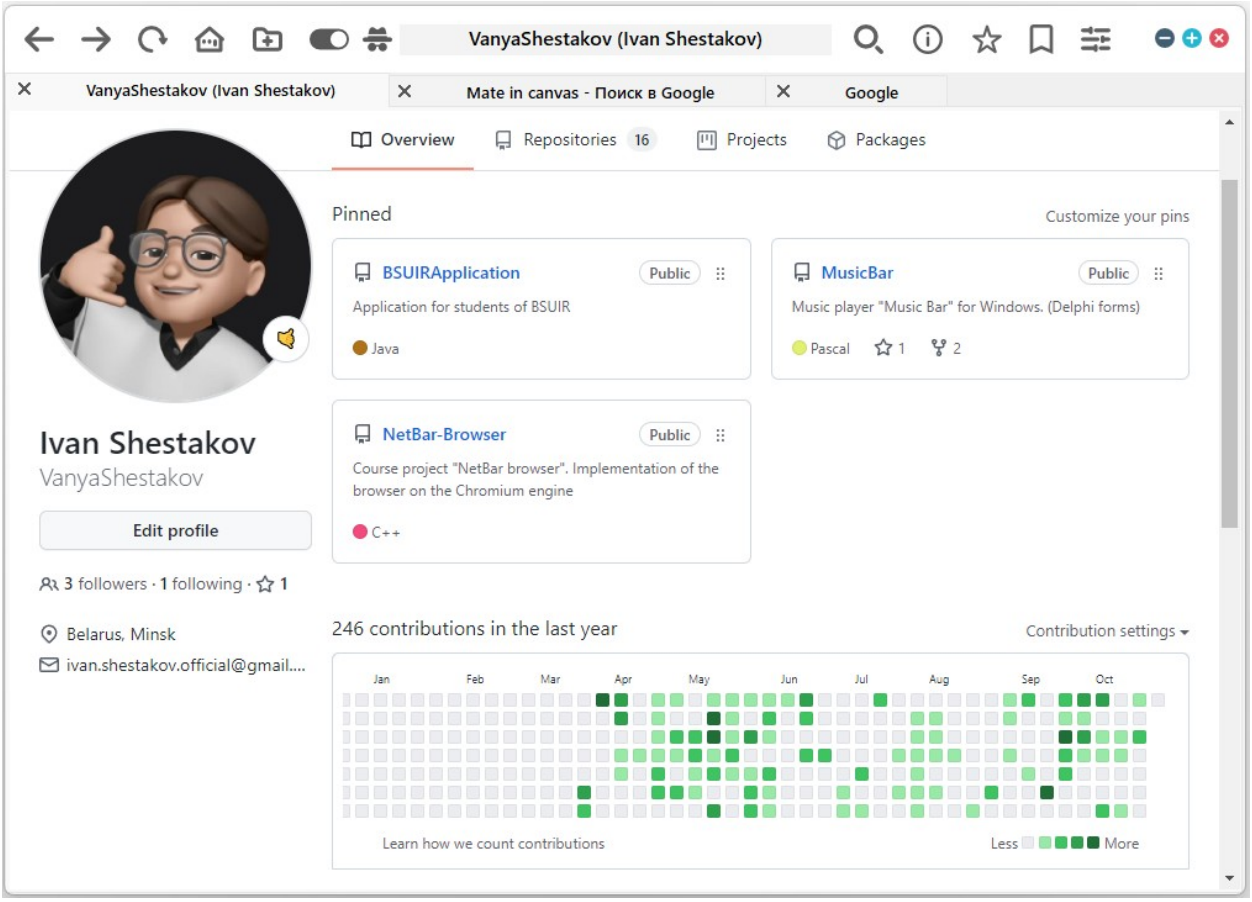

Рисунок 5.1 – Главное окно приложения

5.1.2 Окно настроек

Окно настроек состоит из двух частей. В левой части окна находятся кнопки для выполения различных задач (очистка кэша, очистка данных браузера, очистка закладок, просмотр истории), также предусмотрена возможность смены домашней страницы.

Правая часть окна разворачивается по нажатию кнопки «Показать историю» в левой части окна. В правой части окна находится история посещения сайтов веб-браузера и кнопка для очистки ее очистки . По списку истории можно осуществлять навигацию. Внешний вид окна настроек в свернутом и развернутом виде представлен на рисунке 5.2.

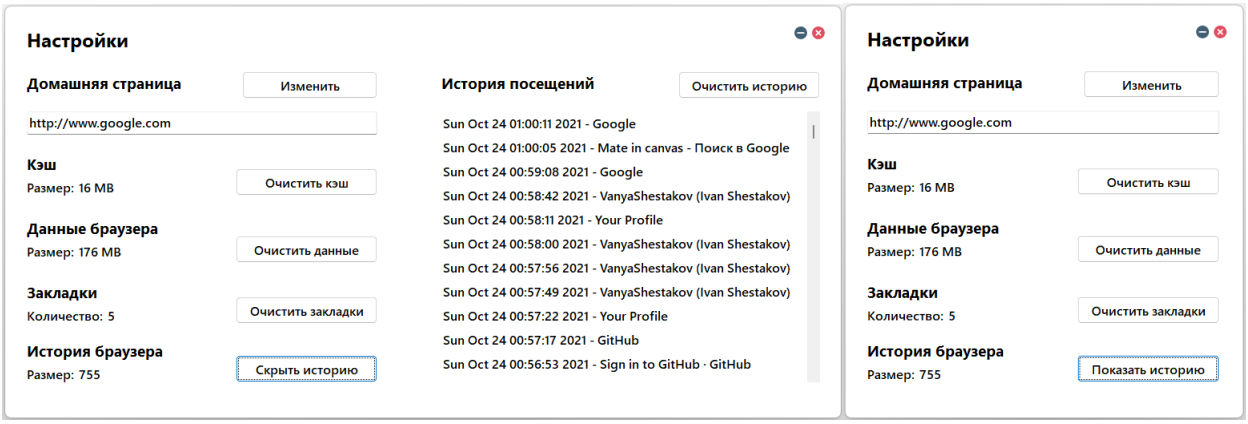

Рисунок 5.2 – Окно настроек

5.1.3 Окно добавления страницы в закладки

Окно добавления страницы в закладки содержит две кнопки для добавления/отмены добавления закладки и поле ввода названия закладки. Внешний вид окна добавления закладок представлен на рисунке 2.5.

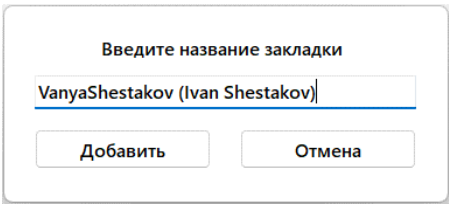

Рисунок 5.3 – Окно добавления закладок

5.1.4 Информационные окна

Программное средство содержит два информационных окна: первое – для отображения информации об анонимном режиме, второе – для отображения информации о программном средстве. Внешний вид информационных окон представлен на рисунке 2.4.

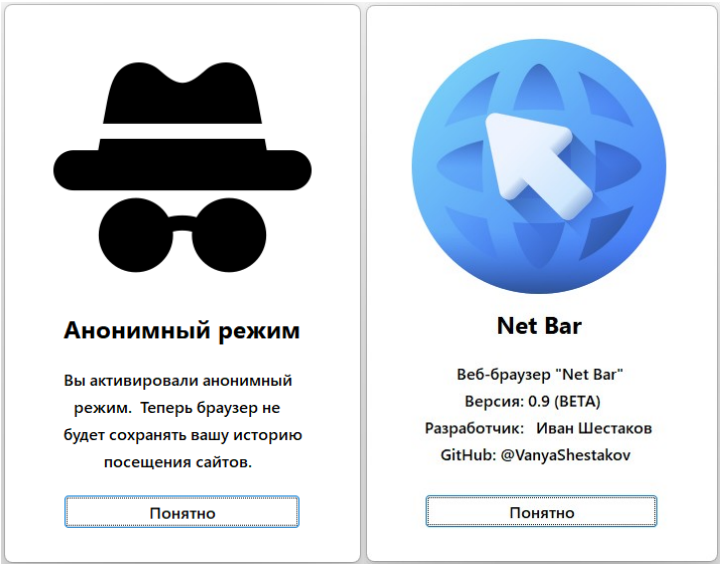

Рисунок 5.4 – Внешний вид информационных окон

5.2 Управление программным средством

5.2.1 Элементы управления веб-браузером

Элементы управления браузером включают в себя различные компоненты такие как кнопки, текстовые поля, выпадающие списки и вкладки. При наведении на каждый элемент управления появляется всплывающая подсказка, говорящая о предназначении компонента. При использовании веб-браузера пользователю доступны следующие функции:

- Возврат на предыдущую страницу;
- Переключение на следующую страницу;
- Обновление текущей страницы;
- Возврат на домашнюю страницу;
- Открытие новой вкладки;
- Закрытие вкладки;
- Включение анонимного режима;
- Поиск запроса в интернете;
- Просмотр информации о приложении;
- Добавление/удаление закладки;
- Просмотр списка закладок;
- Навигация по закладкам;
- Открытие окна настроек;

Внешний вид элементов управления веб-браузером представлен на рисунке 5.5.

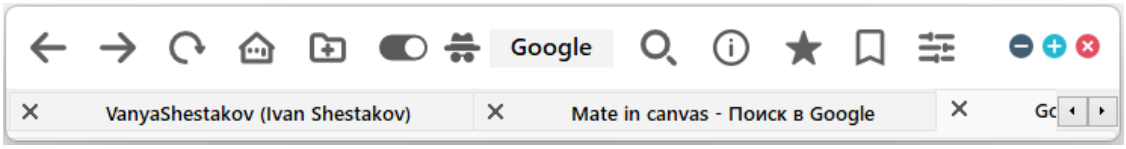

Рисунок 5.5 – Элементы управления веб-браузером

5.2.2 Управление настройками веб-браузера

Управление настройками приложения осуществляется в окне «Настройки». Пользователю доступны следующие возможности в настройках приложения:

- Смена домашней страницы;
- Очистка кэша;
- Очистка всех данных браузера;
- Просмотр информации о занимаемой памяти файлами приложения;
- Очистка закладок;
- Просмотр истории;
- Навигация по истории;
- Очистка истории;

ЗАКЛЮЧЕНИЕ

На сегодняшний день веб-браузеры являются одними из самых популярных приложений на рынке. Людям необходим постоянный выход в интернет поэтому и возрастает потребность в веб-браузерах.

В рамках данного курсового проекта было разработано программное средство «Веб-браузер», которое позволит пользователям операционной системы Windows просматривать сайты в интернете в удобной форме. При разработке программного средства были успешно выполнены все поставленные задачи.

Реализованы базовые функции веб-браузера:

– навигация по сайтам (вперед, назад, возврат на домашнюю страницу);
– обновление текущего сайта;
– поиск в адресной строке;

Также реализованы основные возможности работы со вкладками:

– создание новой вкладки;
– закрытие вкладки;
– переключение между вкладками;

Реализованы базовые возможности для работы с закладками:

– добавление новой закладки;
– удаление закладки;
– очистка всех закладок;
– просмотр всех закладок;

Удалось реализовать функционал, связанный с историей посещения сайтов веб-браузера:

– просмотр истории;
– навигация по истории;
– очистка истории;

Также в приложении были реализованы дополнительные функции:

– «Анонимный режим»;
– возможность очистки кэша браузера;
– возможность очистки всех данных браузера;
– смена домашней страницы.

Существует много способов улучшения данного программного средства. Например, встроить поддержку различных тем оформления веб-браузера. Добавить в приложение поддержку различных расширений. Локализовать данное приложение на несколько популярных языков.

Веб-браузер позволяет людям находится в курсе всего происходящего в мире. Благодаря данному приложению современный мир перевернулся и теперь его трудно представить без веб-браузеров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Microsoft. Microsoft Win32 Programmer’s Reference, 1993. 1000 с.

[2] Симон Р. Windows 95 Win32 Programming Api Bible, 1996. — 1378с.

[3] Петцольд Ч. Programming Windows®, 1998. — СПб.: БХВ-Петербург, 2007. — 1448 с.

[4] Алгоритмы. Теория и практические применение / Род Стивенс. — Москва, 2016 — 544 с.

[5] Мартин Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста. — СПб.: Питер, 2018. — 464 с.

[6] Лафоре Р. Объектно-ориентированное программирование в C++. 4-е полное изд. — СПб.: Питер, 2018. — 928 с.

ПРИЛОЖЕНИЕ А. Исходный код программы

“MainView.h”

```
#ifndef MainViewH
#define MainViewH

#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.Edge.hpp>
#include <WebView2.hpp>
#include <Winapi.ActiveX.hpp>
#include <Vcl.TitleBarCtrls.hpp>
#include <Vcl.ComCtrls.hpp>
#include <Vcl.ToolWin.hpp>
#include <Vcl.WinXCtrls.hpp>
#include <SHDocVw.hpp>
#include <Vcl.OleCtrls.hpp>
#include <Vcl.Buttons.hpp>
#include <Vcl.ExtCtrls.hpp>
#include <Vcl.Graphics.hpp>
#include <Vcl.Imaging.pngimage.hpp>
#include <Vcl.Menus.hpp>
#include <VCLTee.TeCanvas.hpp>
#include <System.ImageList.hpp>
#include <Vcl.ImgList.hpp>
#include <vector>
#include <utility>


#include "PageInfoUnit.h"
#include "BookmarksReader.h"
#include "BookmarksWriter.h"
#include "StringConverter.h"
#include "SettingsView.h"
#include "BookmarksManager.h"
#include "HistoryManager.h"
#include "AddBookmarkView.h"
#include "AnonymModeView.h"
#include "AboutProgramView.h"

class TWebView : public TForm
{
__published:

    TEdit *addressBar;
    TTitleBarPanel *TitleBarPanel1;
    TImage *backBtn;
    TImage *forwardBtn;
    TImage *updateBtn;
    TImage *homeBtn;
    TImage *searchBtn;
    TImage *addBookmarkBtn;
    TImage *settingsBtn;
    TImage *closeBtn;
    TImage *expandBtn;
    TImage *hideBtn;
    TImage *newTabBtn;
```

```
    TPageControl *PageControl;
    TImage *showBookmarksBtn;
    TComboBox *bookmarksBox;
    TPopupMenu *tabsPopup;
    TMenuItem *closeChoice;
    TActivityIndicator *activityIndicator;
    TImage *deleteBookmarkBtn;
    TPopupMenu *bookmarksPopup;
    TMenuItem *N1;
    TImage *disactivateAnonymModeBtn;
    TImage *anonymModeIndicatorBtn;
    TImage *activateAnonymModeBtn;
    TImage *aboutProgramBtn;
    TImageList *ImageList;
    TImage *Image2;
    void __fastcall FormCreate(TObject *Sender);
    void __fastcall backBtnClick(TObject *Sender);
    void __fastcall forwardBtnClick(TObject *Sender);
    void __fastcall updateBtnClick(TObject *Sender);
    void __fastcall searchBtnClick(TObject *Sender);
    void __fastcall backBtnMouseDown(TObject *Sender, TMouseButton Button,
    TShiftState Shift,
          int X, int Y);
    void __fastcall backBtnMouseUp(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y);
    void __fastcall forwardBtnMouseDown(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y);
    void __fastcall forwardBtnMouseUp(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y);
    void __fastcall updateBtnMouseDown(TObject *Sender, TMouseButton Button,
    TShiftState Shift,int X, int Y);
    void __fastcall updateBtnMouseUp(TObject *Sender, TMouseButton Button,
    TShiftState Shift,int X, int Y);
    void __fastcall homeBtnMouseDown(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y);
    void __fastcall homeBtnMouseUp(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y);
    void __fastcall addressBarClick(TObject *Sender);
    void __fastcall homeBtnClick(TObject *Sender);
    void __fastcall searchBtnMouseDown(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y);
    void __fastcall searchBtnMouseUp(TObject *Sender, TMouseButton Button,
    TShiftState Shift, int X, int Y);
       void __fastcall settingsBtnMouseDown(TObject *Sender, TMouseButton Button,
       TShiftState Shift,int X, int Y);
       void __fastcall settingsBtnMouseUp(TObject *Sender, TMouseButton Button,
       TShiftState Shift,int X, int Y);
       void __fastcall closeBtnClick(TObject *Sender);
       void __fastcall closeBtnMouseDown(TObject *Sender, TMouseButton Button,
       TShiftState Shift, int X, int Y);
       void __fastcall closeBtnMouseUp(TObject *Sender, TMouseButton Button,
       TShiftState Shift,int X, int Y);
       void __fastcall hideBtnClick(TObject *Sender);
       void __fastcall hideBtnMouseDown(TObject *Sender, TMouseButton Button,
       TShiftState Shift,int X, int Y);
       void __fastcall hideBtnMouseUp(TObject *Sender, TMouseButton Button,
       TShiftState Shift, int X, int Y);
       void __fastcall expandBtnClick(TObject *Sender);
       void __fastcall addressBarMouseLeave(TObject *Sender);
       void __fastcall newTabBtnClick(TObject *Sender);
```

```
117.      void __fastcall PageControlChange(TObject *Sender);
118.      void __fastcall PageControlMouseUp(TObject *Sender, TMouseButton Button,
119.      TShiftState Shift, int X, int Y);
120.      void __fastcall showBookmarksBtnClick(TObject *Sender);
121.      void __fastcall addBookmarkBtnClick(TObject *Sender);
122.      void __fastcall newTabBtnMouseDown(TObject *Sender, TMouseButton Button,
123.      TShiftState Shift, int X, int Y);
124.      void __fastcall newTabBtnMouseUp(TObject *Sender, TMouseButton Button,
125.      TShiftState Shift, int X, int Y);
126.      void __fastcall showBookmarksBtnMouseDown(TObject *Sender, TMouseButton
127.      Button, TShiftState Shift, int X, int Y);
128.      void __fastcall showBookmarksBtnMouseUp(TObject *Sender, TMouseButton
129.      Button, TShiftState Shift, int X, int Y);
130.      void __fastcall bookmarksBoxSelect(TObject *Sender);
131.      void __fastcall deleteBookmarkBtnClick(TObject *Sender);
132.      void __fastcall settingsBtnClick(TObject *Sender);
133.      void __fastcall closeChoiceClick(TObject *Sender);
134.      void __fastcall FormKeyPress(TObject *Sender, System::WideChar &Key);
135.      void __fastcall activateAnonymModeBtnClick(TObject *Sender);
136.      void __fastcall disactivateAnonymModeBtnClick(TObject *Sender);
137.      void __fastcall anonymModeIndicatorBtnClick(TObject *Sender);
138.      void __fastcall PageControlMouseMove(TObject *Sender, TShiftState Shift,
139.      int X, int Y);
140.      void __fastcall aboutProgramBtnClick(TObject *Sender);
141.
142.  private:
143.
144.      bool isFullScreen = false;
145.      bool isLoaded = true;
146.      bool isSelectedBar = false;
147.      bool isAnonymMode = false;
148.      int tabId = 0;
149.      int tabPopupIndex = 0;
150.      int loadingTab = 0;
151.
152.      const int ANIMATION_OFFSET = 3;
153.      const int TAB_POPUP_OFFSET = 75;
154.      const String NEW_TAB_CAPTION = "Новая вкладка";
155.      const String SEARCH_URL = "https://www.google.com/search?q=";
156.      const String TAB_SPACE = "         ";
157.      const String TAB_NAME_PREFIX = "tab";
158.      const String BROWSER_NAME_POSTFIX = "_browser";
159.      const std::string SECURE_URL_PREFIX = "https://";
160.      const std::string URL_PREFIX = "http://";
161.      const std::string BOOKMARKS_FILENAME = "bookmarks";
162.      const wchar_t* BOOKMARKS_FILE_ACCESS_WARNING = L"Файл с закладками
163.      повреждён, все изменения сохранены не будут";
164.      const wchar_t* MESSAGE_TITLE = L"NetBar";
165.
166.  public:
167.      __fastcall TWebView(TComponent* Owner);
168.      void createNewTab();
169.      TWebBrowser* getCurrentBrowser();
170.      void __fastcall DocumentComplete(TObject *ASender, _di_IDispatch const
171.      pDisp, const OleVariant &URL);
172.      void __fastcall TitleChange(TObject *ASender, const WideString Text);
173.      void __fastcall NewWindow3(TObject *ASender, _di_IDispatch &ppDisp,
174.      WordBool &Cancel, unsigned int val, const WideString bstrUrlContext, const
175.      WideString bstrUrl);
176.      void __fastcall BeforeNavigate2(TObject *ASender, _di_IDispatch const
```

```
177.       pDisp, const OleVariant &URL, const OleVariant &Flags, const OleVariant
178.       &TargetFrameName, const OleVariant &PostData, const OleVariant &Headers,
179.       WordBool &Cancel);
180.       void updateBookmarksBox();
181.       void registerSiteVisit(std::string title, std::string url);
182.       void setAddBookmarkBtnVisible(bool isVisible);
183.       void search(std::string request);
184.       void setLoadingStatus(bool status);
185.       void closeTab(int index);
186.
187.       String title = "";
188.       String pageURL = "";
189.       BookmarksManager *bookmarksManager;
190.       HistoryManager *historyManager;
191.       BrowserManager *browserManager;
192.       StringConverter *converter;
193.       String homepageUrl;
194.
195.   };
196.   extern PACKAGE TWebView *WebView;
197.   #endif
```

“MainView.cpp”

```
198.   #include <vcl.h>
199.   #include <vector>
200.   #include <fstream>
201.   #include <string>
202.   #pragma hdrstop
203.   #include "MainView.h"
204.
205.   #pragma package(smart_init)
206.   #pragma resource "*.dfm"
207.
208.   TWebView *WebView;
209.
210.   __fastcall TWebView::TWebView(TComponent* Owner)
211.       : TForm(Owner)
212.   {
213.       bookmarksManager = new BookmarksManager();
214.       historyManager = new HistoryManager();
215.       browserManager = new BrowserManager();
216.       converter = new StringConverter();
217.       homepageUrl = converter->convertToSystemString(browserManager
218.       ->getHomepageUrl());
219.   }
220.
221.   void __fastcall TWebView::FormCreate(TObject *Sender)
222.   {
223.       createNewTab();
224.       auto bookmarks = bookmarksManager->getBookmarks();
225.       for (int i = 0; i < bookmarks.size(); i++)
226.       {
227.           bookmarksBox
228.               ->Items
229.               ->Add(converter->convertToSystemString(bookmarks[i].first));
230.       }
231.   }
232.
```

```
233.   void __fastcall TWebView::backBtnMouseDown(TObject *Sender, TMouseButton
234.   Button, TShiftState Shift, int X, int Y)
235.   {
236.       backBtn->Top = backBtn->Top + ANIMATION_OFFSET;
237.   }
238.
239.   void __fastcall TWebView::backBtnMouseUp(TObject *Sender, TMouseButton Button,
   TShiftState Shift,
240.             int X, int Y)
241.   {
242.       backBtn->Top = backBtn->Top - ANIMATION_OFFSET;
243.   }
244.
245.   void __fastcall TWebView::forwardBtnMouseDown(TObject *Sender, TMouseButton
   Button,
246.             TShiftState Shift, int X, int Y)
247.   {
248.       forwardBtn->Top = forwardBtn->Top + ANIMATION_OFFSET;
249.   }
250.
251.   void __fastcall TWebView::forwardBtnMouseUp(TObject *Sender, TMouseButton
   Button, TShiftState Shift,
252.             int X, int Y)
253.   {
254.       forwardBtn->Top = forwardBtn->Top - ANIMATION_OFFSET;
255.   }
256.
257.   void __fastcall TWebView::updateBtnMouseDown(TObject *Sender, TMouseButton
   Button, TShiftState Shift,
258.             int X, int Y)
259.   {
260.       updateBtn->Top = updateBtn->Top + ANIMATION_OFFSET;
261.   }
262.
263.   void __fastcall TWebView::updateBtnMouseUp(TObject *Sender, TMouseButton
   Button, TShiftState Shift,
264.             int X, int Y)
265.   {
266.       updateBtn->Top = updateBtn->Top - ANIMATION_OFFSET;
267.   }
268.
269.   void __fastcall TWebView::homeBtnMouseDown(TObject *Sender, TMouseButton
   Button, TShiftState Shift,
270.             int X, int Y)
271.   {
272.       homeBtn->Top = homeBtn->Top + ANIMATION_OFFSET;
273.   }
274.
275.   void __fastcall TWebView::homeBtnMouseUp(TObject *Sender, TMouseButton Button,
   TShiftState Shift,
276.             int X, int Y)
277.   {
278.       homeBtn->Top = homeBtn->Top - ANIMATION_OFFSET;
279.   }
280.
281.   void __fastcall TWebView::searchBtnMouseDown(TObject *Sender, TMouseButton
   Button, TShiftState Shift,
282.             int X, int Y)
283.   {
284.       searchBtn->Top = searchBtn->Top + ANIMATION_OFFSET;
```

```
285.   }
286.
287.   void __fastcall TWebView::searchBtnMouseUp(TObject *Sender, TMouseButton
   Button, TShiftState Shift,
288.             int X, int Y)
289.   {
290.       searchBtn->Top = searchBtn->Top - ANIMATION_OFFSET;
291.   }
292.
293.   void __fastcall TWebView::settingsBtnMouseDown(TObject *Sender, TMouseButton
   Button,
294.             TShiftState Shift, int X, int Y)
295.   {
296.       settingsBtn->Top = settingsBtn->Top + ANIMATION_OFFSET;
297.   }
298.
299.   void __fastcall TWebView::settingsBtnMouseUp(TObject *Sender, TMouseButton
   Button, TShiftState Shift,
300.             int X, int Y)
301.   {
302.       settingsBtn->Top = settingsBtn->Top - ANIMATION_OFFSET;
303.   }
304.
305.   void __fastcall TWebView::newTabBtnMouseDown(TObject *Sender, TMouseButton
   Button,
306.             TShiftState Shift, int X, int Y)
307.   {
308.       newTabBtn->Top = newTabBtn->Top + ANIMATION_OFFSET;
309.   }
310.
311.   void __fastcall TWebView::newTabBtnMouseUp(TObject *Sender, TMouseButton
   Button, TShiftState Shift,
312.             int X, int Y)
313.   {
314.       newTabBtn->Top = newTabBtn->Top - ANIMATION_OFFSET;
315.   }
316.
317.   void __fastcall TWebView::showBookmarksBtnMouseDown(TObject *Sender,
   TMouseButton Button,
318.             TShiftState Shift, int X, int Y)
319.   {
320.       showBookmarksBtn->Top = showBookmarksBtn->Top + ANIMATION_OFFSET;
321.   }
322.
323.   void __fastcall TWebView::showBookmarksBtnMouseUp(TObject *Sender, TMouseButton
   Button,
324.             TShiftState Shift, int X, int Y)
325.   {
326.       showBookmarksBtn->Top = showBookmarksBtn->Top - ANIMATION_OFFSET;
327.   }
328.
329.   void __fastcall TWebView::closeBtnMouseDown(TObject *Sender, TMouseButton
   Button,
330.             TShiftState Shift, int X, int Y)
331.   {
332.       closeBtn->Top = closeBtn->Top + ANIMATION_OFFSET;
333.   }
334.
335.   void __fastcall TWebView::closeBtnMouseUp(TObject *Sender, TMouseButton Button,
   TShiftState Shift,
```

```
336.             int X, int Y)
337.   {
338.       closeBtn->Top = closeBtn->Top - ANIMATION_OFFSET;
339.   }
340.
341.
342.   void __fastcall TWebView::hideBtnMouseDown(TObject *Sender, TMouseButton
   Button, TShiftState Shift,
343.             int X, int Y)
344.   {
345.       hideBtn->Top = hideBtn->Top + ANIMATION_OFFSET;
346.   }
347.
348.   void __fastcall TWebView::hideBtnMouseUp(TObject *Sender, TMouseButton Button,
   TShiftState Shift,
349.             int X, int Y)
350.   {
351.       hideBtn->Top = hideBtn->Top - ANIMATION_OFFSET;
352.   }
353.
354.   void __fastcall TWebView::hideBtnClick(TObject *Sender)
355.   {
356.       Application->Minimize();
357.   }
358.
359.   void __fastcall TWebView::closeBtnClick(TObject *Sender)
360.   {
361.      Application->Terminate();
362.   }
363.
364.
365.   void __fastcall TWebView::expandBtnClick(TObject *Sender)
366.   {
367.       if (isFullScreen)
368.       {
369.           WebView->WindowState = wsNormal;
370.           isFullScreen = false;
371.       }
372.       else
373.       {
374.           WebView->WindowState = wsMaximized;
375.           isFullScreen = true;
376.       }
377.   }
378.
379.   void __fastcall TWebView::backBtnClick(TObject *Sender)
380.   {
381.       getCurrentBrowser()->GoBack();
382.   }
383.
384.   void __fastcall TWebView::forwardBtnClick(TObject *Sender)
385.   {
386.       getCurrentBrowser()->GoForward();
387.   }
388.
389.   void __fastcall TWebView::updateBtnClick(TObject *Sender)
390.   {
391.       getCurrentBrowser()->Refresh();
392.   }
393.
```

```
394.   void __fastcall TWebView::searchBtnClick(TObject *Sender)
395.   {
396.       search(converter->convertToStdString(addressBar->Text));
397.   }
398.   void __fastcall TWebView::homeBtnClick(TObject *Sender)
399.   {
400.       getCurrentBrowser()->Navigate(homepageUrl);
401.   }
402.
403.   void TWebView::search(std::string request)
404.   {
405.       bool isPrefixContains = request.rfind(URL_PREFIX, 0) == 0  ||
406.                               request.rfind(SECURE_URL_PREFIX, 0) == 0;
407.       if (isPrefixContains)
408.       {
409.           getCurrentBrowser()->Navigate(converter-
   >convertToSystemString(request));
410.       }
411.       else
412.       {
413.           getCurrentBrowser()->Navigate(SEARCH_URL + addressBar->Text);
414.       }
415.   }
416.
417.   void __fastcall TWebView::addressBarClick(TObject *Sender)
418.   {
419.       addressBar->Text = pageURL;
420.       if (!isSelectedBar)
421.       {
422.           addressBar->SelectAll();
423.       }
424.       isSelectedBar = true;
425.   }
426.
427.   void __fastcall TWebView::addressBarMouseLeave(TObject *Sender)
428.   {
429.       isSelectedBar = false;
430.   }
431.
432.
433.   void __fastcall TWebView::TitleChange(TObject *ASender, const WideString Text)
434.   {
435.       title = Text;
436.       if (PageControl->ActivePageIndex == loadingTab)
437.       {
438.           addressBar->Text = title;
439.       }
440.       if (!isLoaded) {
441.           PageControl->Pages[loadingTab]->Caption = TAB_SPACE + title +
   TAB_SPACE;
442.       }
443.       else
444.       {
445.           addressBar->Text = title;
446.           PageControl->ActivePage->Caption = TAB_SPACE + title + TAB_SPACE;
447.       }
448.   }
449.
450.   void __fastcall TWebView::DocumentComplete(TObject *ASender, _di_IDispatch
   const pDisp, const OleVariant &URL)
```

```
451.   {
452.       pageURL = URL;
453.       setLoadingStatus(false);
454.       if (!isAnonymMode)
455.       {
456.           registerSiteVisit(converter->convertToStdString(title), converter-
   >convertToStdString(URL));
457.       }
458.       setAddBookmarkBtnVisible(!bookmarksManager->contains(converter-
   >convertToStdString(pageURL)));
459.   }
460.
461.   void TWebView::registerSiteVisit(std::string title, std::string url)
462.   {
463.       time_t now = time(0);
464.       std::string stringTime = ctime(&now);
465.       historyManager->addSiteVisit(stringTime, title, url);
466.   }
467.
468.   void TWebView::setAddBookmarkBtnVisible(bool isVisible)
469.   {
470.       deleteBookmarkBtn->Visible = !isVisible;
471.       addBookmarkBtn->Visible = isVisible;
472.   }
473.
474.   void __fastcall TWebView::NewWindow3(TObject *ASender, _di_IDispatch &ppDisp,
   WordBool &Cancel,
475.             unsigned int val, const WideString bstrUrlContext, const WideString
   bstrUrl)
476.   {
477.       createNewTab();
478.       getCurrentBrowser()->Navigate(bstrUrl);
479.       Cancel = true;
480.   }
481.
482.   void __fastcall TWebView::newTabBtnClick(TObject *Sender)
483.   {
484.       if (isLoaded)
485.       {
486.           createNewTab();
487.       }
488.   }
489.
490.   void TWebView::createNewTab()
491.   {
492.       TTabSheet *tab = new TTabSheet(PageControl);
493.       tab->Caption = NEW_TAB_CAPTION;
494.       tab->PageControl = PageControl;
495.       PageControl->ActivePage = tab;
496.       tab->Name = TAB_NAME_PREFIX + String(tabId);
497.
498.       TWebBrowser *browser = new TWebBrowser(PageControl);
499.       browser->SelectedEngine = TWebBrowser::TSelectedEngine::EdgeOnly;
500.       browser->TOleControl::Parent = tab;
501.       browser->Width = tab->Width;
502.       browser->Height = tab->Height;
503.       browser->Align = alClient;
504.       browser->Navigate(homepageUrl);
505.       tab->Tag = Integer(browser);
506.       browser->TOleControl::Name = tab->Name + BROWSER_NAME_POSTFIX;
```

```
507.
508.        browser->OnDocumentComplete = DocumentComplete;
509.        browser->OnNewWindow3 = NewWindow3;
510.        browser->OnTitleChange = TitleChange;
511.        browser->OnBeforeNavigate2 = BeforeNavigate2;
512.
513.        tabId++;
514.    }
515.
516.    void TWebView::closeTab(int index)
517.    {
518.        PageControl->Pages[index]->Free();
519.        setLoadingStatus(false);
520.        if (index != PageControl->PageCount)
521.        {
522.            PageControl->ActivePageIndex = index;
523.        }
524.        else
525.        {
526.            PageControl->ActivePageIndex = index - 1;
527.        }
528.        title = getCurrentBrowser()->LocationName;
529.        pageURL = getCurrentBrowser()->LocationURL;
530.        addressBar->Text = title;
531.        setAddBookmarkBtnVisible(!bookmarksManager->contains(converter-
    >convertToStdString(pageURL)));
532.    }
533.
534.    void __fastcall TWebView::PageControlChange(TObject *Sender)
535.    {
536.        title = getCurrentBrowser()->LocationName;
537.        pageURL = getCurrentBrowser()->LocationURL;
538.        addressBar->Text = title;
539.        setAddBookmarkBtnVisible(!bookmarksManager->contains(converter-
    >convertToStdString(pageURL)));
540.    }
541.
542.    void TWebView::setLoadingStatus(bool status)
543.    {
544.        isLoaded = !status;
545.        if (status)
546.        {
547.            activityIndicator->StartAnimation();
548.        }
549.        else
550.        {
551.            activityIndicator->StopAnimation();
552.        }
553.        updateBtn->Visible = !status;
554.        activityIndicator->Visible = status;
555.    }
556.
557.    void __fastcall TWebView::PageControlMouseUp(TObject *Sender, TMouseButton
    Button,
558.              TShiftState Shift, int X, int Y)
559.    {
560.        const int CLOSE_BTN_SIZE = 20;
561.        if (Button == mbLeft)
562.        {
```

```
563.           TRect tabRect = PageControl->TabRect(PageControl->ActivePage-
   >TabIndex);
564.
565.           bool isClose = PtInRect(
566.               Classes::Rect(tabRect.Left, tabRect.Top,
567.               tabRect.Left + CLOSE_BTN_SIZE, tabRect.Top + CLOSE_BTN_SIZE),
568.               Classes::Point(X, Y));
569.
570.           if (isClose && PageControl->PageCount > 1)
571.           {
572.               closeTab(PageControl->IndexOfTabAt(X, Y));
573.           }
574.       }
575.   }
576.
577.   TWebBrowser* TWebView::getCurrentBrowser()
578.   {
579.       return (TWebBrowser*) PageControl
580.           ->ActivePage
581.           ->FindChildControl(PageControl->ActivePage->Name +
   BROWSER_NAME_POSTFIX);
582.   }
583.
584.   void __fastcall TWebView::showBookmarksBtnClick(TObject *Sender)
585.   {
586.       bookmarksBox->DroppedDown = true;
587.   }
588.
589.   void __fastcall TWebView::addBookmarkBtnClick(TObject *Sender)
590.   {
591.       if (isLoaded)
592.       {
593.           AddBookmarkForm->ShowModal();
594.       }
595.   }
596.
597.   void TWebView::updateBookmarksBox()
598.   {
599.       bookmarksBox->Clear();
600.       auto bookmarks = bookmarksManager->getBookmarks();
601.       for (int i = 0; i < bookmarks.size(); i++)
602.       {
603.           bookmarksBox
604.               ->Items
605.               ->Add(converter->convertToSystemString(bookmarks[i].first));
606.       }
607.   }
608.
609.   void __fastcall TWebView::BeforeNavigate2(TObject *ASender, _di_IDispatch const
   pDisp,
610.             const OleVariant &URL, const OleVariant &Flags, const OleVariant
   &TargetFrameName,
611.             const OleVariant &PostData, const OleVariant &Headers,
612.             WordBool &Cancel)
613.   {
614.       loadingTab = PageControl->ActivePageIndex;
615.       setLoadingStatus(true);
616.   }
617.
618.   void __fastcall TWebView::bookmarksBoxSelect(TObject *Sender)
```

```
619.   {
620.       auto bookmarks = bookmarksManager->getBookmarks();
621.       String url = converter->convertToSystemString(bookmarks[bookmarksBox-
   >ItemIndex].second);
622.       getCurrentBrowser()->Navigate(url);
623.   }
624.
625.   void __fastcall TWebView::deleteBookmarkBtnClick(TObject *Sender)
626.   {
627.       if (isLoaded)
628.       {
629.           bool isOpenFile = bookmarksManager->removeBookmark(converter-
   >convertToStdString(pageURL));
630.           if (!isOpenFile)
631.           {
632.               Application
633.                   ->MessageBox(BOOKMARKS_FILE_ACCESS_WARNING,
634.                           MESSAGE_TITLE,
635.                           MB_OK | MB_ICONWARNING);
636.           }
637.           updateBookmarksBox();
638.           addBookmarkBtn->Visible = true;
639.           deleteBookmarkBtn->Visible = false;
640.       }
641.   }
642.
643.   void __fastcall TWebView::settingsBtnClick(TObject *Sender)
644.   {
645.       SettingsForm->ShowModal();
646.   }
647.
648.   void __fastcall TWebView::closeChoiceClick(TObject *Sender)
649.   {
650.       PageControl->Pages[tabPopupIndex]->Free();
651.       title = getCurrentBrowser()->LocationName;
652.       pageURL = getCurrentBrowser()->LocationURL;
653.       addressBar->Text = title;
654.       setAddBookmarkBtnVisible(!bookmarksManager->contains(converter-
   >convertToStdString(pageURL)));
655.   }
656.
657.   void __fastcall TWebView::FormKeyPress(TObject *Sender, System::WideChar &Key)
658.   {
659.       if(Key == 13)
660.       {
661.           search(converter->convertToStdString(addressBar->Text));
662.       }
663.   }
664.
665.   void __fastcall TWebView::activateAnonymModeBtnClick(TObject *Sender)
666.   {
667.       activateAnonymModeBtn->Visible = false;
668.       disactivateAnonymModeBtn->Visible = true;
669.       anonymModeIndicatorBtn->Visible = true;
670.       isAnonymMode = true;
671.   }
672.
673.   void __fastcall TWebView::disactivateAnonymModeBtnClick(TObject *Sender)
674.   {
675.       activateAnonymModeBtn->Visible = true;
```

```
676.       disactivateAnonymModeBtn->Visible = false;
677.       anonymModeIndicatorBtn->Visible = false;
678.       isAnonymMode = false;
679.   }
680.
681.   void __fastcall TWebView::anonymModeIndicatorBtnClick(TObject *Sender)
682.   {
683.       AnonymModeForm->ShowModal();
684.   }
685.
686.   void __fastcall TWebView::PageControlMouseMove(TObject *Sender, TShiftState
   Shift,
687.             int X, int Y)
688.   {
689.       TRect R = PageControl->TabRect(PageControl->IndexOfTabAt(X, Y));
690.       if (PtInRect(Classes::Rect(R.Left, R.Top, R.Left + 20, R.Top + 20),
   Classes::Point(X, Y)))
691.       {
692.           PageControl->Cursor = crHandPoint;
693.       }
694.       else
695.       {
696.           PageControl->Cursor = crDefault;
697.       }
698.   }
699.
700.   void __fastcall TWebView::aboutProgramBtnClick(TObject *Sender)
701.   {
702.       AboutProgramForm->ShowModal();
703.   }
```

“SettingsView.h”

```
704.   #ifndef SettingsViewH
705.   #define SettingsViewH
706.
707.   #include <System.Classes.hpp>
708.   #include <Vcl.Controls.hpp>
709.   #include <Vcl.StdCtrls.hpp>
710.   #include <Vcl.Forms.hpp>
711.   #include <Vcl.TitleBarCtrls.hpp>
712.   #include <Vcl.ExtCtrls.hpp>
713.   #include <Vcl.Imaging.pngimage.hpp>
714.   #include <Vcl.Buttons.hpp>
715.   #include "BrowserManager.h"
716.   #include "MainView.h"
717.
718.   class TSettingsForm : public TForm
719.   {
720.
721.   __published:
722.       TImage *hideBtn;
723.       TImage *closeBtn;
724.       TLabel *settingsTitle;
725.       TLabel *casheTitle;
726.       TLabel *casheSizeTitle;
727.       TLabel *casheSizeLabel;
728.       TBitBtn *clearCasheBtn;
```

```
729.       TLabel *browserDataTitle;
730.       TLabel *browserDataSizeTitle;
731.       TLabel *browserDataSizeLabel;
732.       TBitBtn *clearBrowserDataBtn;
733.       TLabel *browserHistoryTitle;
734.       TLabel *browserHistorySizeTitle;
735.       TLabel *browserHistorySizeLabel;
736.       TBitBtn *clearBrowserHistoryBtn;
737.       TLabel *bookmarksTitle;
738.       TLabel *bookmarksAmountTitle;
739.       TLabel *Label5;
740.       TBitBtn *clearBookmarksBtn;
741.       TLabel *bookmarksAmountLabel;
742.       TListBox *historyBox;
743.       TLabel *Label3;
744.       TEdit *homepageUrlEdit;
745.       TBitBtn *changeHomepageBtn;
746.       TLabel *Label1;
747.       TBitBtn *showHistoryBtn;
748.       void __fastcall closeBtnClick(TObject *Sender);
749.       void __fastcall hideBtnClick(TObject *Sender);
750.       void __fastcall FormShow(TObject *Sender);
751.       void __fastcall clearCasheBtnClick(TObject *Sender);
752.       void __fastcall clearBrowserDataBtnClick(TObject *Sender);
753.       void __fastcall clearBookmarksBtnClick(TObject *Sender);
754.       void __fastcall clearBrowserHistoryBtnClick(TObject *Sender);
755.       void __fastcall showHistoryBtnClick(TObject *Sender);
756.       void __fastcall historyBoxDblClick(TObject *Sender);
757.       void __fastcall changeHomepageBtnClick(TObject *Sender);
758.
759.   private:
760.
761.       BrowserManager *browserManager;
762.       bool isOpenedHistory;
763.       const int KB_SIZE = 1024;
764.       const int EXTENDED_HEIGHT = 840;
765.       const int DEFAULT_HEIGHT = 600;
766.       const int EXTENDED_WIDTH = 1050;
767.       const int DEFAULT_WIDTH = 500;
768.       const wchar_t* MESSAGE_TITLE = L"NetBar";
769.       const wchar_t* RESTART_WARNING = L"Для того, чтобы изменения вступили в
   силу, приложение сейчас перезапустится";
770.       const wchar_t* BROWSER_CLEANING_WARNING = L"Очистка данных браузера
   приведёт к удалению всех паролей, cookie файлов, кэша и др. После очистки
   приложение будет перезапущено. Вы действительно хотите очистить все данные
   браузера?";
771.       const wchar_t* BOOKMARKS_CLEANING_WARNING = L"Вы действительно хотите
   удалить все закладки безвозвратно";
772.       const wchar_t* HOMEPAGE_WARNING = L"Внутренние файлы повреждены, изменения
   сохранены не будут";
773.       const wchar_t* HOMEPAGE_SUCCESS = L"Домашняя страница успешно обновлена";
774.       const String SHOW_HISTORY_CAPTION = "Показать историю";
775.       const String HIDE_HISTORY_CAPTION = "Скрыть историю";
776.
777.   public:
778.
779.       __fastcall TSettingsForm(TComponent* Owner);
780.       void updateHistoryBox();
781.
782.   };
```

```
783.   extern PACKAGE TSettingsForm *SettingsForm;
784.   #endif
785.
```

“SettingsView.cpp”

```
786.
787.   #include <vcl.h>
788.   #pragma hdrstop
789.
790.   #include "SettingsView.h"
791.   #include "SiteVisit.h"
792.   #include <vector>
793.
794.   #pragma resource "*.dfm"
795.   TSettingsForm *SettingsForm;
796.   __fastcall TSettingsForm::TSettingsForm(TComponent* Owner)
797.       : TForm(Owner)
798.   {
799.       browserManager = new BrowserManager();
800.   }
801.
802.   void __fastcall TSettingsForm::closeBtnClick(TObject *Sender)
803.   {
804.       SettingsForm->Close();
805.   }
806.
807.   void __fastcall TSettingsForm::hideBtnClick(TObject *Sender)
808.   {
809.       Application->Minimize();
810.   }
811.
812.   void __fastcall TSettingsForm::FormShow(TObject *Sender)
813.   {
814.       int casheSize = browserManager->getCasheSize();
815.       int browserDataSize = browserManager->getBrowserDataSize();
816.       casheSizeLabel->Caption = IntToStr((casheSize / KB_SIZE / KB_SIZE)) + "
   MB";
817.       browserDataSizeLabel->Caption = IntToStr((browserDataSize / KB_SIZE /
   KB_SIZE)) + " MB";
818.       bookmarksAmountLabel->Caption = IntToStr(WebView->bookmarksManager-
   >getSize());
819.       browserHistorySizeLabel->Caption = IntToStr(WebView->historyManager-
   >getSize());
820.       updateHistoryBox();
821.       isOpenedHistory = false;
822.       homepageUrlEdit->Text = WebView->homepageUrl;
823.       SettingsForm->Width = DEFAULT_WIDTH;
824.       showHistoryBtn->Caption = SHOW_HISTORY_CAPTION;
825.   }
826.
827.   void __fastcall TSettingsForm::clearCasheBtnClick(TObject *Sender)
828.   {
829.       browserManager->clearCashe();
830.       Application
831.           ->MessageBox(RESTART_WARNING,
832.                        MESSAGE_TITLE,
833.                        MB_OK | MB_ICONINFORMATION);
834.       ShellExecute(0, 0, Application->ExeName.c_str(), 0, 0, SW_SHOW);
```

```
835.        Application->Terminate();
836.    }
837.
838.    void __fastcall TSettingsForm::clearBrowserDataBtnClick(TObject *Sender)
839.    {
840.        if (Application->MessageBox(BROWSER_CLEANING_WARNING, MESSAGE_TITLE,
841.            MB_YESNO | MB_ICONINFORMATION) == IDYES)
842.        {
843.            delete WebView->PageControl;
844.            Sleep(200);
845.            browserManager->clearBrowserData();
846.            ShellExecute(0, 0, Application->ExeName.c_str(), 0, 0, SW_SHOW);
847.            Application->Terminate();
848.        }
849.    }
850.
851.    void __fastcall TSettingsForm::clearBookmarksBtnClick(TObject *Sender)
852.    {
853.        if (Application->MessageBox(BOOKMARKS_CLEANING_WARNING, MESSAGE_TITLE,
854.            MB_YESNO | MB_ICONINFORMATION) == IDYES)
855.        {
856.            WebView->bookmarksManager->clearBookmarks();
857.            WebView->bookmarksBox->Clear();
858.            bookmarksAmountLabel->Caption = IntToStr(WebView->bookmarksManager-
    >getSize());
859.        }
860.    }
861.
862.    void TSettingsForm::updateHistoryBox()
863.    {
864.        std::vector<SiteVisit> history = WebView->historyManager->getHistory();
865.        historyBox->Clear();
866.        for (int i = history.size() - 1; i >= 0; i--)
867.        {
868.            historyBox->Items->Add((history[i].getTime() + " - " +
869.                                    history[i].getTitle()).c_str());
870.        }
871.    }
872.
873.    void __fastcall TSettingsForm::clearBrowserHistoryBtnClick(TObject *Sender)
874.    {
875.        WebView->historyManager->clearHistory();
876.        historyBox->Clear();
877.        browserHistorySizeLabel->Caption = IntToStr(WebView->historyManager-
    >getSize());
878.    }
879.
880.    void __fastcall TSettingsForm::showHistoryBtnClick(TObject *Sender)
881.    {
882.        if (isOpenedHistory)
883.        {
884.            isOpenedHistory = false;
885.            SettingsForm->Width = DEFAULT_WIDTH;
886.            showHistoryBtn->Caption = HIDE_HISTORY_CAPTION;
887.        }
888.        else
889.        {
890.            isOpenedHistory = true;
891.            SettingsForm->Width = EXTENDED_WIDTH;
892.            showHistoryBtn->Caption = SHOW_HISTORY_CAPTION;
```

```
893.       }
894.   }
895.
896.   void __fastcall TSettingsForm::historyBoxDblClick(TObject *Sender)
897.   {
898.       int size = WebView->historyManager->getSize();
899.       int index = size - historyBox->ItemIndex - 1;
900.       std::vector<SiteVisit> history = WebView->historyManager->getHistory();
901.       WebView->createNewTab();
902.       WebView->getCurrentBrowser()->Navigate(history[index].getUrl().c_str());
903.       SettingsForm->Close();
904.   }
905.
906.   void __fastcall TSettingsForm::changeHomepageBtnClick(TObject *Sender)
907.   {
908.       StringConverter *converter = new StringConverter();
909.       String url = homepageUrlEdit->Text;
910.       bool isOpen = WebView->browserManager->setHomepageUrl(converter-
   >convertToStdString(url));
911.       WebView->homepageUrl = url;
912.       if (isOpen)
913.       {
914.           Application
915.           ->MessageBox(HOMEPAGE_SUCCESS,
916.                        MESSAGE_TITLE,
917.                        MB_OK | MB_ICONINFORMATION);
918.       }
919.       else
920.       {
921.           Application
922.           ->MessageBox(HOMEPAGE_WARNING,
923.                        MESSAGE_TITLE,
924.                        MB_OK | MB_ICONINFORMATION);
925.       }
926.   }
927.
```

“AddBookmarkView.h”

```
928.   #ifndef AddBookmarkViewH
929.   #define AddBookmarkViewH
930.
931.   #include <System.Classes.hpp>
932.   #include <Vcl.Controls.hpp>
933.   #include <Vcl.StdCtrls.hpp>
934.   #include <Vcl.Forms.hpp>
935.   #include <Vcl.Buttons.hpp>
936.   #include <Vcl.ExtCtrls.hpp>
937.   #include <Vcl.Imaging.pngimage.hpp>
938.
939.   #include <utility>
940.   #include <string>
941.   #include <regex>
942.   #include "MainView.h"
943.
944.   class TAddBookmarkForm : public TForm
945.   {
946.   __published:
947.
```

```
948.       TEdit *titleEdit;
949.       TBitBtn *addBtn;
950.       TBitBtn *cancelBtn;
951.       TLabel *casheSizeTitle;
952.       void __fastcall cancelBtnClick(TObject *Sender);
953.       void __fastcall addBtnClick(TObject *Sender);
954.       void __fastcall FormShow(TObject *Sender);
955.       void __fastcall titleEditChange(TObject *Sender);
956.
957.   private:
958.
959.       const wchar_t* BOOKMARKS_FILE_ACCESS_WARNING = L"Файл с закладками
    повреждён, все изменения сохранены не будут";
960.       const wchar_t* MESSAGE_TITLE = L"NetBar";
961.
962.   public:
963.
964.       __fastcall TAddBookmarkForm(TComponent* Owner);
965.
966.   };
967.   extern PACKAGE TAddBookmarkForm *AddBookmarkForm;
968.   #endif
```

“AddBookmarkView.cpp”

```
969.   #include <vcl.h>
970.   #pragma hdrstop
971.
972.   #include "AddBookmarkView.h"
973.
974.   #pragma package(smart_init)
975.   #pragma resource "*.dfm"
976.
977.   TAddBookmarkForm *AddBookmarkForm;
978.
979.   __fastcall TAddBookmarkForm::TAddBookmarkForm(TComponent* Owner)
980.       : TForm(Owner)
981.   {
982.
983.   }
984.
985.   void __fastcall TAddBookmarkForm::cancelBtnClick(TObject *Sender)
986.   {
987.       AddBookmarkForm->Close();
988.   }
989.
990.   void __fastcall TAddBookmarkForm::addBtnClick(TObject *Sender)
991.   {
992.       StringConverter *converter = new StringConverter();
993.
994.       bool isOpenFile = WebView->bookmarksManager
995.           ->addBookmark(converter->convertToStdString(titleEdit->Text.Trim()),
996.                         converter->convertToStdString(WebView->pageURL));
997.       delete converter;
998.       if (!isOpenFile)
999.       {
1000.          Application
1001.          ->MessageBox(BOOKMARKS_FILE_ACCESS_WARNING,
1002.                       MESSAGE_TITLE,
```

```
                      MB_OK | MB_ICONWARNING);
    }
    WebView->updateBookmarksBox();
    WebView->addBookmarkBtn->Visible = false;
    WebView->deleteBookmarkBtn->Visible = true;
    AddBookmarkForm->Close();
}

void __fastcall TAddBookmarkForm::FormShow(TObject *Sender)
{
    titleEdit->Text = WebView->title;
}

void __fastcall TAddBookmarkForm::titleEditChange(TObject *Sender)
{
    StringConverter *converter = new StringConverter();
    std::string title = converter->convertToStdString(titleEdit->Text.Trim());
    addBtn->Enabled =
        !WebView->bookmarksManager->titleExists(title) &&
        title != "";
}
```

“BookmarksManager.h”

```
#ifndef BookmarksManagerH
#define BookmarksManagerH
#include <vector>
#include <utility>
#include <fstream>
#include <regex>

#include "BookmarksReader.h"
#include "BookmarksWriter.h"
#include "StringConverter.h"

class BookmarksManager
{
    private:

        std::vector<std::pair<std::string, std::string>> bookmarks;
        BookmarksWriter *writer;
        BookmarksReader *reader;
        void readBookmarks(std::string path);
        bool writeBookmarks(std::string path);
        const std::string FILENAME = "bookmarks";

    public:

        BookmarksManager();
        ~BookmarksManager();
        bool addBookmark(std::string title, std::string url);
        bool contains(std::string url);
        bool titleExists(std::string title);
        bool removeBookmark(std::string url);
        bool clearBookmarks();
        std::vector<std::pair<std::string, std::string>> getBookmarks();
        int getSize();

};
```

```
#endif
```

“BookmarksManager.cpp”

```
#pragma hdrstop

#include "BookmarksManager.h"

#pragma package(smart_init)

BookmarksManager::BookmarksManager()
{
    writer = new BookmarksWriter();
    reader = new BookmarksReader();
    readBookmarks(FILENAME);
}

BookmarksManager::~BookmarksManager()
{
    delete writer;
    delete reader;
    std::vector<std::pair<std::string, std::string>>().swap(bookmarks);
}

void BookmarksManager::readBookmarks(std::string path)
{
    bookmarks = reader->readBookmarks(path);
}

bool BookmarksManager::writeBookmarks(std::string path)
{
    return writer->writeBookmarks(bookmarks, path);
}

bool BookmarksManager::addBookmark(std::string title, std::string url)
{
    std::pair<std::string, std::string> pair;
    pair.first = title;
    pair.second = url;
    bookmarks.push_back(pair);
    return writeBookmarks(FILENAME);
}

bool BookmarksManager::clearBookmarks()
{
    bookmarks.clear();
    return writeBookmarks(FILENAME);
}

bool BookmarksManager::contains(std::string url)
{
    for (int i = 0; i < bookmarks.size(); ++i)
    {
       if (bookmarks[i].second == url)
       {
           return true;
       }
    }
    return false;
```

```
1115.  }
1116.
1117.  bool BookmarksManager::titleExists(std::string title)
1118.  {
1119.      for (int i = 0; i < bookmarks.size(); ++i)
1120.      {
1121.         if (bookmarks[i].first == title)
1122.         {
1123.             return true;
1124.         }
1125.      }
1126.      return false;
1127.  }
1128.
1129.  bool BookmarksManager::removeBookmark(std::string url)
1130.  {
1131.      int index = 0;
1132.      while (index < bookmarks.size() && url != bookmarks[index].second)
1133.      {
1134.          index++;
1135.      }
1136.      if (index != bookmarks.size()) {
1137.          bookmarks.erase(bookmarks.begin() + index);
1138.          return writeBookmarks(FILENAME);
1139.      }
1140.      return false;
1141.  }
1142.
1143.  std::vector<std::pair<std::string, std::string>>
     BookmarksManager::getBookmarks()
1144.  {
1145.      return bookmarks;
1146.  }
1147.
1148.  int BookmarksManager::getSize()
1149.  {
1150.      return bookmarks.size();
1151.  }
```

“HistoryManager.h”

```
1152.  #ifndef HistoryManagerH
1153.  #define HistoryManagerH
1154.
1155.  #include <vector>
1156.  #include <utility>
1157.  #include <fstream>
1158.  #include <regex>
1159.
1160.  #include "HistoryReader.h"
1161.  #include "HistoryWriter.h"
1162.  #include "StringConverter.h"
1163.  #include "SiteVisit.h"
1164.
1165.  class HistoryManager
1166.  {
1167.      private:
1168.
1169.          std::vector<SiteVisit> history;
```

```
1170.          HistoryWriter *writer;
1171.          HistoryReader *reader;
1172.          void readHistory(std::string path);
1173.          bool writeHistory(std::string path);
1174.          const std::string FILENAME = "history";
1175.
1176.      public:
1177.
1178.          HistoryManager();
1179.          ~HistoryManager();
1180.          bool addSiteVisit(std::string time, std::string title, std::string
   url);
1181.          bool clearHistory();
1182.          std::vector<SiteVisit> getHistory();
1183.          int getSize();
1184.
1185.  };
1186.
1187.  #endif
1188.
```

“HistoryManager.cpp”

```
1189.  #pragma hdrstop
1190.
1191.  #include "HistoryManager.h"
1192.
1193.  #pragma package(smart_init)
1194.
1195.  HistoryManager::HistoryManager()
1196.  {
1197.      writer = new HistoryWriter();
1198.      reader = new HistoryReader();
1199.      readHistory(FILENAME);
1200.  }
1201.
1202.  HistoryManager::~HistoryManager()
1203.  {
1204.      delete writer;
1205.      delete reader;
1206.      std::vector<SiteVisit>().swap(history);
1207.  }
1208.
1209.  void HistoryManager::readHistory(std::string path)
1210.  {
1211.      history = reader->readHistory(path);
1212.  }
1213.
1214.  bool HistoryManager::writeHistory(std::string path)
1215.  {
1216.      return writer->writeHistory(history, path);
1217.  }
1218.
1219.  bool HistoryManager::addSiteVisit(std::string time, std::string title,
   std::string url)
1220.  {
1221.      SiteVisit *visit = new SiteVisit(title, url, time);
1222.      history.push_back(*visit);
1223.      return writeHistory(FILENAME);
```

```
}

bool HistoryManager::clearHistory()
{
    history.clear();
    return writeHistory(FILENAME);
}

std::vector<SiteVisit> HistoryManager::getHistory()
{
    return history;
}

int HistoryManager::getSize()
{
    return history.size();
}
```

“BrowserManager.h”

```
#ifndef BrowserManagerH
#define BrowserManagerH

#include <System.Classes.hpp>
#include <utility>
#include <filesystem>
#include <fstream>
#include <string>
#include <ctype.h>

class BrowserManager
{
    private:

        const AnsiString CASHE_PATH =
"Project1.exe.WebView2\\EBWebView\\Default\\Cache";
        const AnsiString BROWSER_DATA_PATH =
"Project1.exe.WebView2\\EBWebView\\Default";
        const std::string DEFAULT_HOMEPAGE_URL = "https://www.google.com/";
        const std::string HOMEPAGE_FILENAME = "homepage";
        std::string homepageUrl;
        int getFolderSize(AnsiString folder, int &size);
        std::string readHomePageUrl(std::string path);
        bool writeHomepageUrl(std::string path, std::string newUrl);

    public:

        BrowserManager();
        int getCasheSize();
        std::string getHomepageUrl();
        void clearCashe();
        int getBrowserDataSize();
        void clearBrowserData();
        bool setHomepageUrl(std::string newUrl);
};
#endif
```

“BrowserManager.cpp”

```
#pragma hdrstop

#include "BrowserManager.h"

#pragma package(smart_init)

BrowserManager::BrowserManager()
{
    homepageUrl = readHomePageUrl(HOMEPAGE_FILENAME);
    if (homepageUrl.empty())
    {
        homepageUrl = DEFAULT_HOMEPAGE_URL;
    }
}

int BrowserManager::getFolderSize(AnsiString folder, int &size)
{
    TSearchRec searchRec;
    if (folder[folder.Length()] == '\\')
    {
        folder.SetLength(folder.Length() - 1);
    }

    if (FindFirst(folder + "\\*.*", faAnyFile, searchRec) == 0)
    {
        do
        {
            if (searchRec.Name != "." && searchRec.Name != "..")
            {
                if ((searchRec.Attr & faDirectory) != 0)
                {
                    getFolderSize(folder + "\\" + searchRec.Name, size);
                }
                else
                {
                    size += searchRec.Size;
                }
            }
        } while (FindNext(searchRec) == 0);
    }
    FindClose(searchRec);
    return size;
}

int BrowserManager::getCasheSize()
{
    int size = 0;
    return getFolderSize(CASHE_PATH, size);
}

std::string BrowserManager::getHomepageUrl()
{
    return homepageUrl;
}

int BrowserManager::getBrowserDataSize()
{
```

```
1335.      int size = 0;
1336.      return getFolderSize(BROWSER_DATA_PATH, size);
1337.  }
1338.
1339.  void BrowserManager::clearCashe()
1340.  {
1341.      std::filesystem::remove_all(CASHE_PATH.c_str());
1342.  }
1343.
1344.  void BrowserManager::clearBrowserData()
1345.  {
1346.      std::filesystem::remove_all(BROWSER_DATA_PATH.c_str());
1347.  }
1348.
1349.  std::string BrowserManager::readHomePageUrl(std::string path)
1350.  {
1351.      std::ifstream fileReader;
1352.      std::string url;
1353.      fileReader.open(path);
1354.      if (fileReader.is_open())
1355.      {
1356.          std::getline(fileReader, url);
1357.          fileReader.close();
1358.      }
1359.      return url;
1360.  }
1361.
1362.  bool BrowserManager::setHomepageUrl(std::string newUrl)
1363.  {
1364.      homepageUrl = newUrl;
1365.      return writeHomepageUrl(HOMEPAGE_FILENAME, newUrl);
1366.  }
1367.
1368.  bool BrowserManager::writeHomepageUrl(std::string path, std::string newUrl)
1369.  {
1370.      std::ofstream fileWriter;
1371.      std::string url;
1372.      fileWriter.open(path);
1373.      bool isOpen = fileWriter.is_open();
1374.      if (isOpen)
1375.      {
1376.          fileWriter << newUrl;
1377.      }
1378.      return isOpen;
1379.  }
1380.
1381.  #ifndef BookmarksReaderH
1382.  #define BookmarksReaderH
1383.
1384.  #include <vector>
1385.  #include <utility>
1386.  #include <fstream>
1387.  #include <regex>
```

“BookmarksReader.h”

```
1388.  class BookmarksReader
1389.  {
1390.      private:
1391.
1392.      public:
1393.          std::vector<std::pair<std::string, std::string>>
   readBookmarks(std::string path);
1394.  };
1395.  #endif
```

“BookmarksReader.cpp”

```
1396.  #pragma hdrstop
1397.
1398.  #include "BookmarksReader.h"
1399.
1400.  #pragma package(smart_init)
1401.
1402.  std::vector<std::pair<std::string, std::string>>
   BookmarksReader::readBookmarks(std::string path)
1403.  {
1404.      const std::regex r(R"(^.+\S=.+\S$)");
1405.      std::vector<std::pair<std::string, std::string>> bookmarks;
1406.      std::ifstream fileReader;
1407.      std::string line;
1408.      fileReader.open(path);
1409.      if (fileReader.is_open())
1410.      {
1411.          while (!fileReader.eof())
1412.          {
1413.              std::pair<std::string, std::string> pair;
1414.              std::getline(fileReader, line);
1415.              if (!std::regex_match(line, r))
1416.              {
1417.                  bookmarks.clear();
1418.                  return bookmarks;
1419.              }
1420.              int index = line.find("=");
1421.              pair.first = line.substr(0, index);
1422.              pair.second = line.substr(index + 1);
1423.              bookmarks.push_back(pair);
1424.          }
1425.          fileReader.close();
1426.      }
1427.      return bookmarks;
1428.  }
```

“BookmarksWriter.h”

```
1429.  #ifndef BookmarksWriterH
1430.  #define BookmarksWriterH
1431.
1432.  #include <vector>
1433.  #include <utility>
1434.  #include <fstream>
1435.
```

```
1436.  class BookmarksWriter
1437.  {
1438.      private:
1439.
1440.      public:
1441.          bool writeBookmarks(std::vector<std::pair<std::string, std::string>>
   bookmarks, std::string path);
1442.  };
1443.  #endif
```

“BookmarksWriter.cpp”

```
1444.  #pragma hdrstop
1445.
1446.  #include "BookmarksWriter.h"
1447.
1448.  #pragma package(smart_init)
1449.
1450.  bool BookmarksWriter::writeBookmarks(std::vector<std::pair<std::string,
   std::string>> bookmarks, std::string path)
1451.  {
1452.      std::ofstream writer;
1453.      std::string line;
1454.      writer.open(path);
1455.      bool isOpen = writer.is_open();
1456.      if (isOpen)
1457.      {
1458.          for (int i = 0; i < bookmarks.size(); ++i)
1459.          {
1460.              if(i + 1 == bookmarks.size())
1461.              {
1462.                  writer << bookmarks[i].first + "=" + bookmarks[i].second;
1463.              }
1464.              else
1465.              {
1466.                  writer << bookmarks[i].first + "=" + bookmarks[i].second <<
   std::endl;
1467.              }
1468.          }
1469.          writer.close();
1470.      }
1471.      return isOpen;
1472.  }
```

“HistoryReader.h”

```
1473.  #ifndef HistoryReaderH
1474.  #define HistoryReaderH
1475.
1476.  #include "SiteVisit.h"
1477.  #include <vector>
1478.  #include <utility>
1479.  #include <fstream>
1480.  #include <regex>
1481.
1482.  class HistoryReader
1483.  {
1484.      private:
1485.
```

```
    public:
        std::vector<SiteVisit> readHistory(std::string path);
};

#endif
```

“HistoryReader.cpp”

```
#pragma hdrstop

#include "HistoryReader.h"

#pragma package(smart_init)

std::vector<SiteVisit> HistoryReader::readHistory(std::string path)
{
    std::vector<SiteVisit> history;
    std::ifstream fileReader;
    std::string line;
    std::string time;
    std::string title;
    std::string url;
    fileReader.open(path);
    if (fileReader.is_open())
    {
        while (!fileReader.eof())
        {
            std::getline(fileReader, time);
            if (!fileReader.eof())
            {
                std::getline(fileReader, title);
            }
            else
            {
                history.clear();
                return history;
            }

            if (!fileReader.eof())
            {
                std::getline(fileReader, url);
            }
            else
            {
                history.clear();
                return history;
            }
            SiteVisit *visit = new SiteVisit(title, url, time + "\n");
            history.push_back(*visit);
        }
        fileReader.close();
    }
    return history;
}

```

“HistoryWriter.h”

```
#ifndef HistoryWriterH
#define HistoryWriterH

#include "SiteVisit.h"
#include <vector>
#include <utility>
#include <fstream>

class HistoryWriter
{
    private:

    public:

        bool writeHistory(std::vector<SiteVisit> history, std::string path);
};
#endif
```

“HistoryWriter.cpp”

```
#pragma hdrstop

#include "HistoryWriter.h"

#pragma package(smart_init)

bool HistoryWriter::writeHistory(std::vector<SiteVisit> history, std::string
   path)
{
    std::ofstream writer;
    std::string line;
    writer.open(path);
    bool isOpen = writer.is_open();
    if (isOpen)
    {
        for (int i = 0; i < history.size(); ++i)
        {
            if(i + 1 == history.size())
            {
                writer << history[i].getTime();
                writer << history[i].getTitle() << std::endl;
                writer << history[i].getUrl();
            }
            else
            {
                writer << history[i].getTime();
                writer << history[i].getTitle() << std::endl;
                writer << history[i].getUrl() << std::endl;
            }
        }
        writer.close();
    }
    return isOpen;
}

```

“SiteVisit.h”

```
#ifndef SiteVisitH
#define SiteVisitH

#include <string>

class SiteVisit
{
    private:

        std::string title;
        std::string url;
        std::string time;

    public:

        SiteVisit(std::string title, std::string url, std::string time);
        std::string getTitle();
        std::string getUrl();
        std::string getTime();
};

#endif
```

“SiteVisit.cpp”

```
#pragma hdrstop

#include "SiteVisit.h"

#pragma package(smart_init)

SiteVisit::SiteVisit(std::string title, std::string url, std::string time)
{
    this->title = title;
    this->url = url;
    this->time = time;
}

std::string SiteVisit::getTitle()
{
    return this->title;
}

std::string SiteVisit::getUrl()
{
    return this->url;
}
std::string SiteVisit::getTime()
{
    return this->time;
}

```

“StringConverter.h”

```
#ifndef StringConverterH
#define StringConverterH

#include <string>
#include <System.Classes.hpp>

class StringConverter
{
    private:

    public:

        std::string convertToStdString(String str);
        String convertToSystemString(std::string str);
};
#endif
```

“StringConverter.cpp”

```
#pragma hdrstop

#include "StringConverter.h"

#pragma package(smart_init)

std::string StringConverter::convertToStdString(String str)
{
    AnsiString ansiStr = str;
    return ansiStr.c_str();
}

String StringConverter::convertToSystemString(std::string str)
{
    return str.c_str();
}
```