Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

Направление подготовки 11.03.02

Лабораторная работа №2

Использование Git и Gulp
для решения задач web-разработки

Выполнил:

Швалов Даниил Андреевич

Группа: К33211

Проверила:

Марченко Елена Вадимовна

Санкт-Петербург

2023

1. Введение

Цель работы: научиться использовать систему контроля версий Git для отслеживания и ведения истории изменения файлов в проектах, познакомиться с инструментом для автоматизации, организации и обработке задач Gulp, разработать собственное приложение для просмотра веб-страниц.

2. Ход работы

Задание №1

В данном задании необходимо установить Git на компьютер, настроить на работу с проектом, выполнить изменения в файлах проекта. Для выполняемых изменений сделать коммиты (не менее трех). Проверить, что коммиты создаются. Локальный репозиторий синхронизировать с удаленным.

В качестве файлов, которые будут отслеживаться системой контроля версий Git, были выбраны файлы проектов всех лабораторных работ по веб-программированию. С помощью команды `git init`, выполненной в корне проекта, был инициализирован репозиторий. С помощью команды `git add` все интересующие файлы были помечены как отслеживаемые. После этого была использована команда `git commit -m "message"` для фиксации изменений. Вместо `message` были использованы сообщения, описывающие изменения в данной фиксации.

После проделанных действий было необходимо синхронизировать локальный репозиторий с удаленным. В качестве удаленного репозитория был выбран GitHub, поскольку это популярная площадка для хранения кода, обладающая богатым функционалом для разработчиков. Для GitHub существует множество руководств, что позволяет быстро найти ответ на нужный вопрос. В добавок ко всему, GitHub позволяет бесплатно создавать сколь угодно большое количество публичных репозиториев.

Для того, чтобы загрузить локальный репозиторий на GitHub, был создан удаленный репозиторий на GitHub. После этого была выполнена следующая команда:

```
git remote add origin \
  https://github.com/danilshvalov/itmo-web-programming.git
```

Данная команда устанавливает ссылку на удаленный репозиторий, с которым будет синхронизирован локальный репозиторий. Чтобы отправить изменения на удаленный репозиторий была выполнена команда `git push origin main`, которая синхронизирует текущие изменения в ветке `main`.

В итоге все файлы, которые были помечены как отслеживаемые, были синхронизированы с удаленным репозиторием (рис. 1). На рис. 2 показана история фиксаций, отображаемая в интерфейсе GitHub.

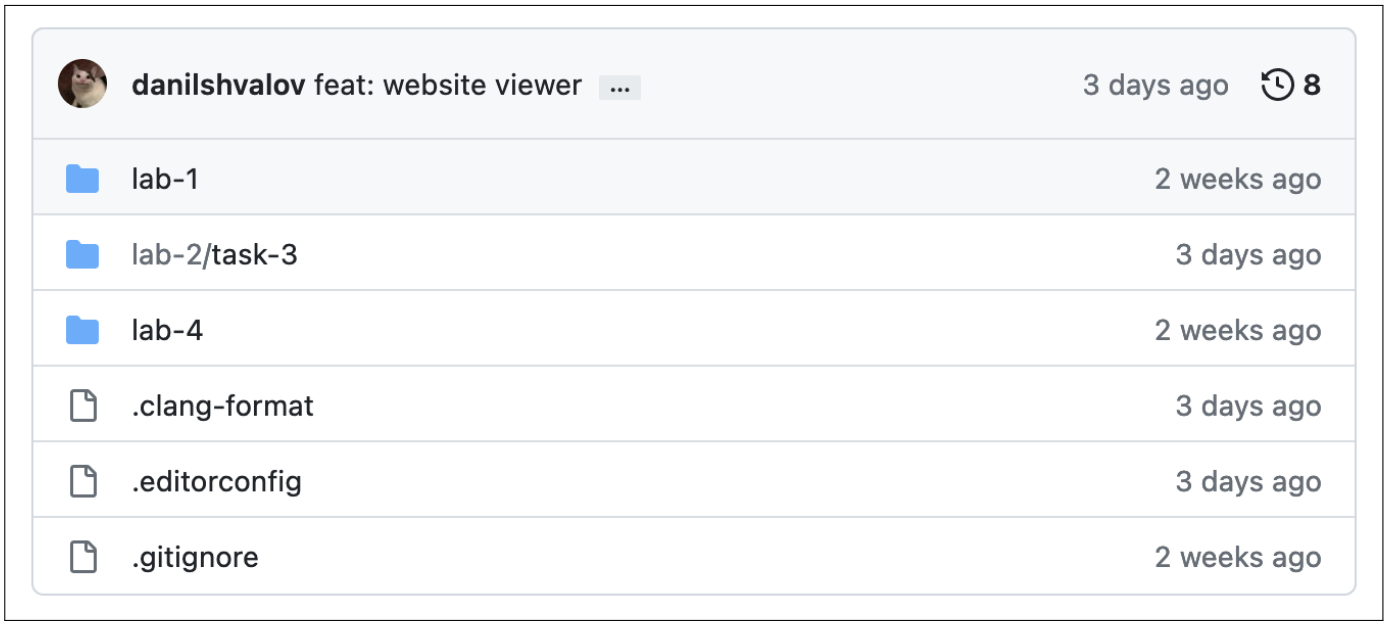

Рисунок 1 – Файлы на удаленном репозитории GitHub

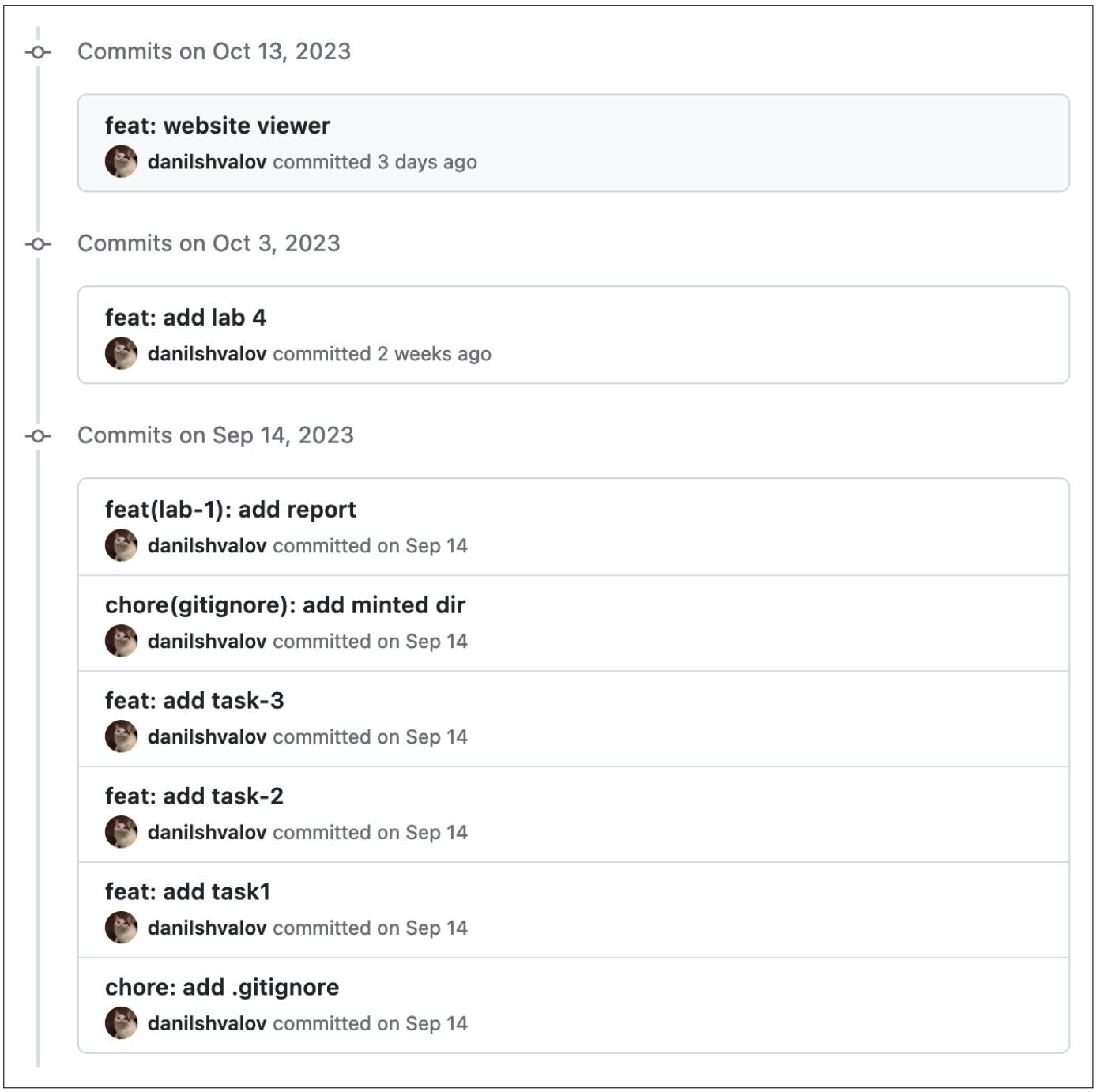

Рисунок 2 – История фиксаций в интерфейсе GitHub

Задание №2

В данном задании необходимо установить Gulp, отметив этапы установки, а также создать какой-нибудь task.

Для выполнения задания был инициализирован проект с помощью команды `npm init`. После выполнения команды в корне проекта появился файл `package.json`, ко-

торый содержит в себе информацию о проекте. Содержимое этого файла находится в приложении А.

После этого в проект был установлен Gulp с помощью команды `npm i gulp`. Также с помощью команды `npm i browser-sync` был установлен BrowserSync — утилита, которая автоматически перезагружает измененные файлы и страницы, синхронизирует навигацию между браузерами, а также позволяет тестировать сайт сразу на нескольких устройствах.

После установки в корне проекта был создан файл `gulpfile.js`, исходный код которого находится в приложении Б. Данный файл создает task с названием `browserSync`, который отслеживает изменения файлов и обновляет браузер в случае, если файлы были изменены. Кроме того, был создан task с названием `default`, чтобы было можно запускать Gulp не указывая в аргументах название task. Также для тестирования Gulp был создан файл `index.html`, исходный код которого находится в приложении В.

Для запуска Gulp необходимо выполнить команду `gulp` в эмуляторе терминала. При выполнении этой команды запустится сервер, который будет отслеживать изменения файлов (рис. 3 и 4).

```
task-2 > gulp
[21:46:00] Using gulpfile ~/projects/web-programming/labs/lab-2/task-2/gulpfile.
[21:46:00] Starting 'default'...
[21:46:00] Starting 'browserSync'...
[Browsersync] Access URLs:
 --------------------------------------
       Local: http://localhost:3000
    External: http://192.168.1.105:3000
 --------------------------------------
          UI: http://localhost:3001
 UI External: http://localhost:3001
 --------------------------------------
[Browsersync] Serving files from: ./
```

Рисунок 3 – Вывод Gulp после запуска

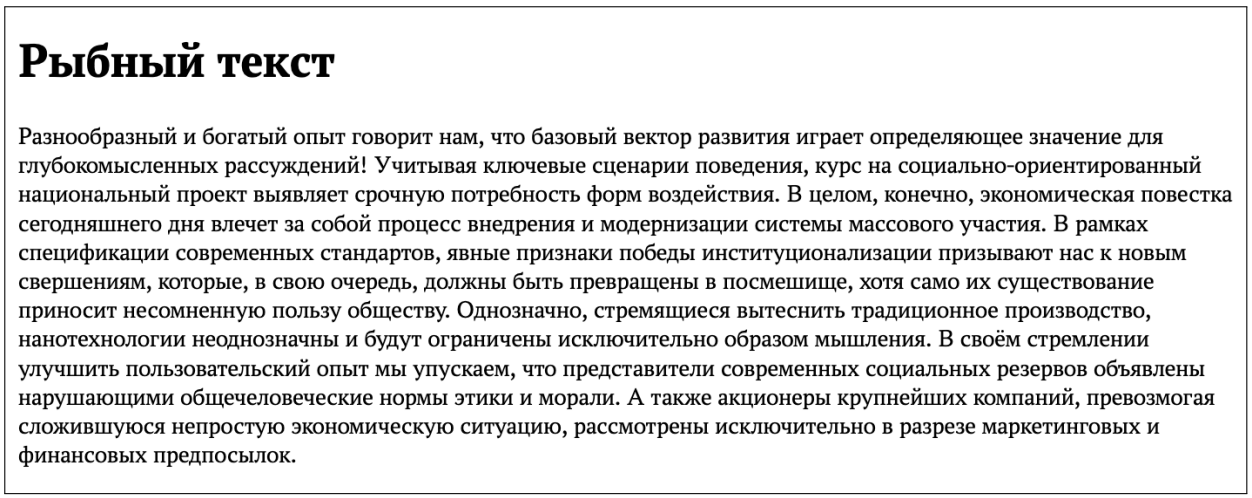

Рисунок 4 – Страница браузера до изменения

При изменении содержимого `index.html` браузер автоматически перезагрузится с новым содержимым (рис. 5).

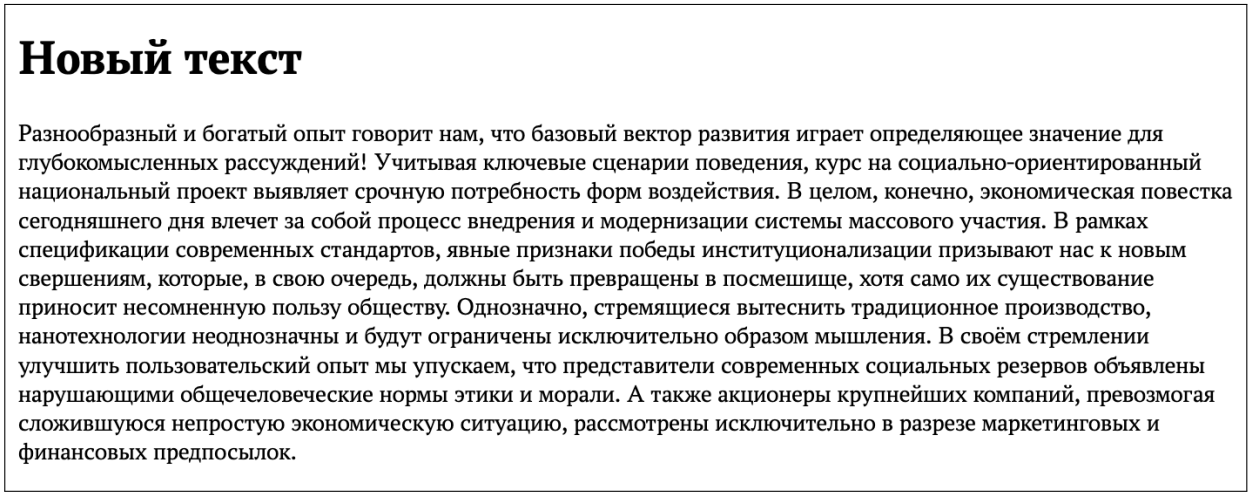

Рисунок 5 – Страница браузера после изменения

Задание №3

В данном задании необходимо написать программу клиент, которая показывает web-страницы одна за другой из списка, при этом в программе можно задавать адреса страниц и интервал показа.

Для решения данного задания был выбран следующий стек технологий. В качестве языка программирования был выбран язык C++, поскольку это компилируемый, статически типизированный язык программирования общего назначения. Он

отличается высокой производительностью, а также переносимостью между системами и платформами. В добавок ко всему, для этого языка существует множество различных библиотек, что позволяет использовать C++ в самых различных сферах деятельности.

В качестве библиотеки, с помощью которой будет создаваться графический пользовательский интерфейс, был выбран фреймворк Qt. Данный фреймворк является очень популярным решением для разработки графических пользовательских интерфейсов среди языков C++ и Python. Для него существует множество материалов, с помощью которых можно найти ответ практически на любой вопрос. Также этот фреймворк поддерживает все основные операционные системы, что позволяет запускать приложения, основанные на этом фреймворке, почти повсюду.

На рис. 6 изображено главное окно программы. На нем отображаются все ссылки, добавленные пользователем. Также имеется возможность создания, редактирования и удаления уже существующих ссылок.

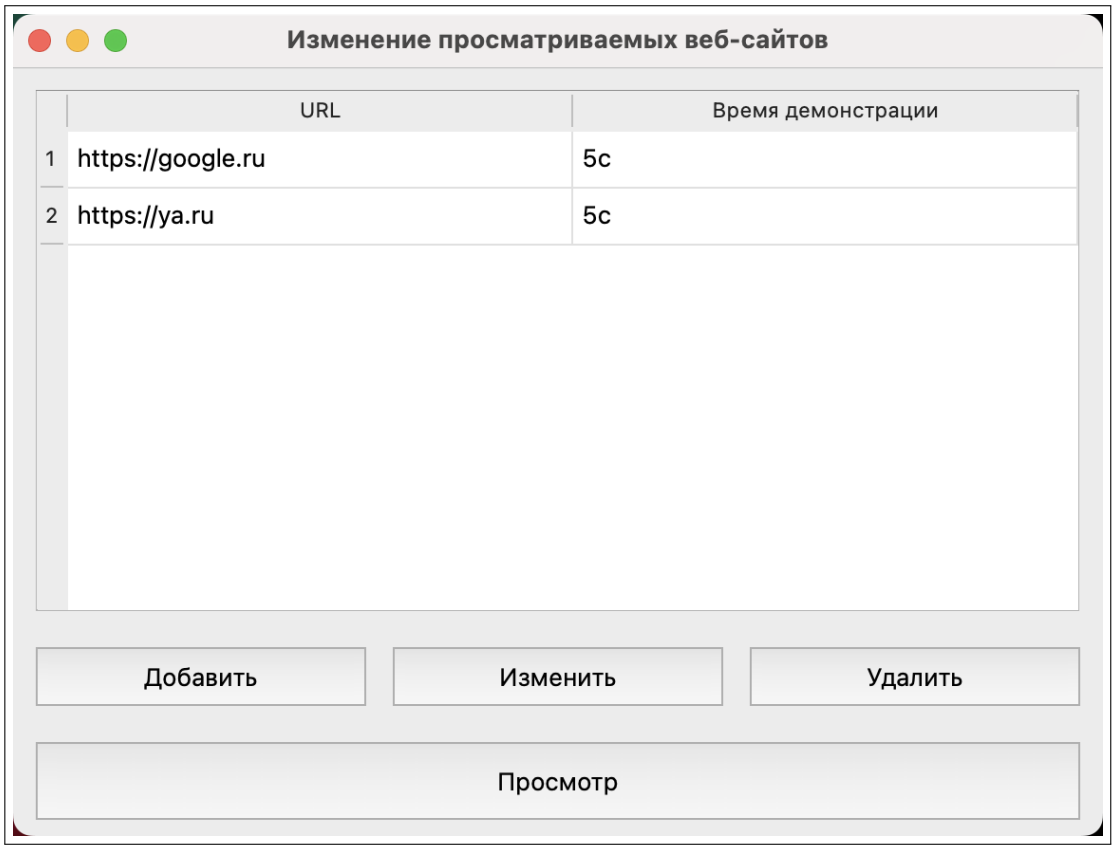

Рисунок 6 – Главное окно программы

При нажатии на кнопку «Добавить» пользователю открывается окно, показанное на рис. 7. В данном окне пользователь должен указать ссылку, которая будет показана, а также время демонстрации сайта по этой ссылке. Ссылка должна быть корректной, иначе кнопка «Сохранить» не станет активной.

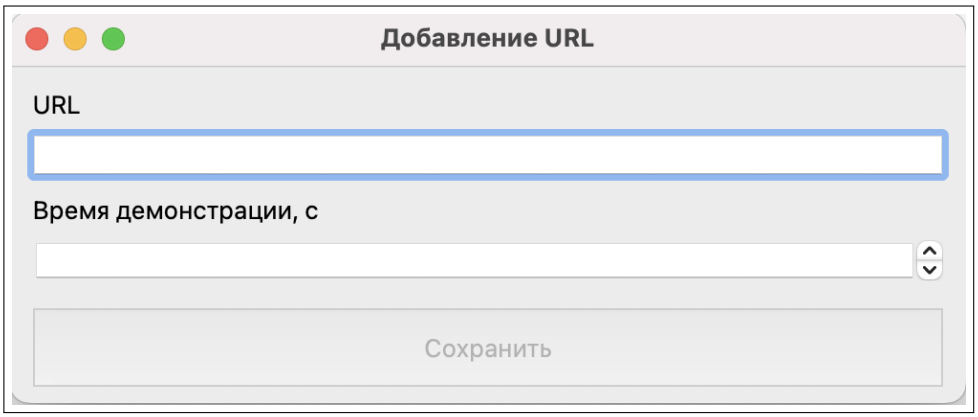

Рисунок 7 – Окно добавления новой ссылки

При двойном нажатии на ссылку или при нажатии на кнопку «Изменить», открывается окно, изображенное на рис. 8. В данном окне пользователь может изменить ссылку, которая будет показана, а также время демонстрации сайта по этой ссылке.

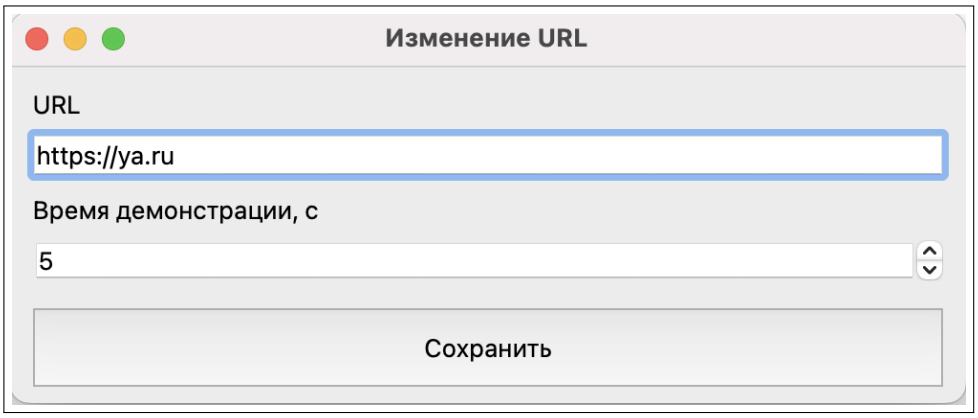

Рисунок 8 – Окно изменения существующей ссылки

При нажатии кнопки «Удалить» выбранная ссылка удаляется без какого-либо дополнительного окна. При нажатии кнопки «Просмотр» открывается окно демонстрации веб-сайтов, изображенное на рис. 9. В нем показывается ссылка, оставшееся

время демонстрации, а также сама демонстрируемая страница. После того, как время выйдет, будет показана следующая ссылка. Если ссылка была последней, окно демонстрации будет закрыто.

Исходный код приложения, а также скриптов сборки, находится в приложениях Г-С, а также доступен в репозитории на GitHub.

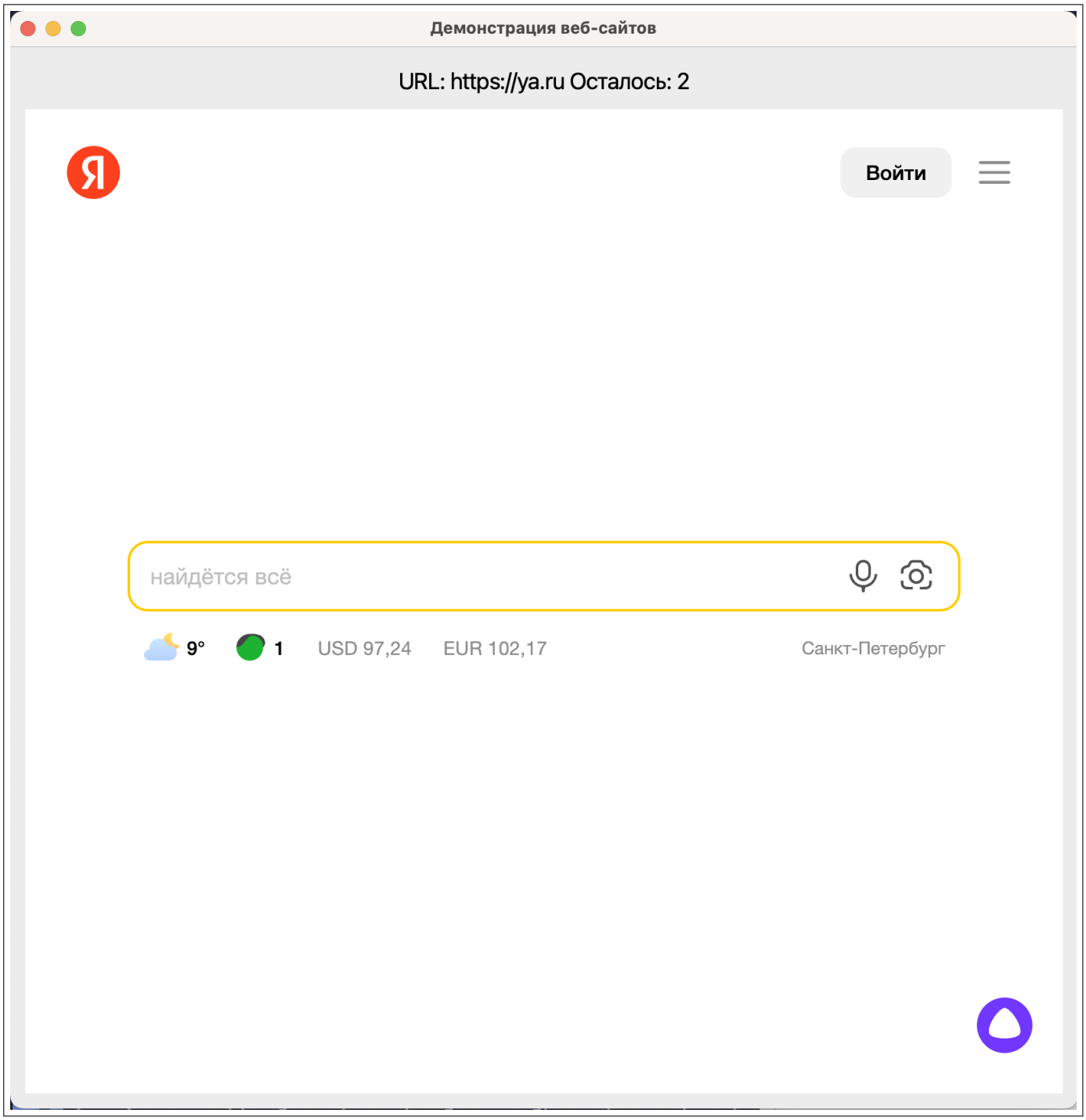

Рисунок 9 – Окно демонстрации веб-сайтов

3. Заключение

Вывод: в данной лабораторной работе я научился использовать систему контроля версий Git для отслеживания и ведения истории изменения файлов в проектах, познакомился с инструментом для автоматизации, организации и обработке задач Gulp, разработал собственное приложение просмотра веб-страниц.

Приложение А

Исходный код package.json

```
{
  "name": "task-2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Приложение Б

Исходный код gulpfile.js

```
const gulp = require("gulp");
const browserSync = require("browser-sync").create();

gulp.task("browserSync", function () {
  browserSync.init({
    server: {
      baseDir: "./",
    },
  });

  gulp.watch("*.html").on("change", browserSync.reload);
});

gulp.task("default", gulp.series("browserSync"));
```

Приложение В

Исходный код index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Gulp Browsersync</title>
  </head>
  <body>
    <h1>Рыбный текст</h1>
    <p>
      Разнообразный и богатый опыт говорит нам, что базовый вектор развития
      играет определяющее значение для глубокомысленных рассуждений! Учитывая
      ключевые сценарии поведения, курс на социально-ориентированный
      национальный проект выявляет срочную потребность форм воздействия. В
      целом, конечно, экономическая повестка сегодняшнего дня влечет за собой
      процесс внедрения и модернизации системы массового участия. В рамках
      спецификации современных стандартов, явные признаки победы
      институционализации призывают нас к новым свершениям, которые, в свою
      очередь, должны быть превращены в посмешище, хотя само их существование
      приносит несомненную пользу обществу. Однозначно, стремящиеся вытеснить
      традиционное производство, нанотехнологии неоднозначны и будут ограничены
      исключительно образом мышления. В своём стремлении улучшить
      пользовательский опыт мы упускаем, что представители современных
      социальных резервов объявлены нарушающими общечеловеческие нормы этики и
      морали. А также акционеры крупнейших компаний, превозмогая сложившуюся
      непростую экономическую ситуацию, рассмотрены исключительно в разрезе
      маркетинговых и финансовых предпосылок.
    </p>
  </body>
</html>
```

Приложение Г

Исходный код CMakeLists.txt

```
cmake_minimum_required(VERSION 3.20)
project(main)

set(CMAKE_AUTOMOC ON)
set(CMAKE_INCLUDE_CURRENT_DIR ON)

find_package(Qt6 REQUIRED
  Core
  PrintSupport
  Gui
  Widgets
  WebEngineWidgets
)

add_executable(
  main
  src/main.cpp
  src/website_viewer.hpp
  src/website_viewer.cpp
  src/table_list_model.hpp
  src/website_list_model.hpp
  src/website_list_model.cpp
  src/website_list_window.hpp
  src/website_list_window.cpp
  src/website_form.hpp
  src/website_form.cpp
  src/url_validator.hpp
  src/url_validator.cpp
)

target_link_libraries(main PUBLIC
  Qt6::Widgets
  Qt6::PrintSupport
  Qt6::WebEngineWidgets
)
```

Приложение Д

Исходный код main.cpp

```
#include "website_form.hpp"
#include "website_list_window.hpp"

#include <QApplication>
#include <QModelIndex>
#include <QTableWidget>
#include <QtWebEngineWidgets/QtWebEngineWidgets>

int main(int argc, char* argv[]) {
    QApplication app(argc, argv);

    WebsiteListWindow* website_window = new WebsiteListWindow();

    website_window->resize(800, 600);
    website_window->show();

    return app.exec();
}
```

Приложение Е

Исходный код website_form.hpp

```
#pragma once

#include "url_validator.hpp"
#include "website_display_info.hpp"

#include <QLabel>
#include <QLineEdit>
#include <QMainWindow>
#include <QPushButton>
#include <QSpinBox>

class WebsiteForm : public QMainWindow {
    Q_OBJECT

   public:
    WebsiteForm(QWidget* parent = nullptr);

    void setFields(WebsiteDisplayInfo info);

    int getRow() const;

    void setRow(int row);

    void clear();

   signals:
    void submit(WebsiteDisplayInfo info);

   private:
    void onInput();

    void onSubmit(bool);

    QLabel* url_label_;
    QLineEdit* url_input_;
    UrlValidator* url_validator_;
```

```
    QLabel* show_time_label_;
    QSpinBox* show_time_input_;
    QPushButton* submit_button_;
    int row_ = -1;
};
```

Приложение Ж

Исходный код website_form.cpp

```
#include "website_form.hpp"

#include <QVBoxLayout>

WebsiteForm::WebsiteForm(QWidget* parent) : QMainWindow(parent) {
    url_label_ = new QLabel(this);
    url_label_->setText("URL");

    show_time_label_ = new QLabel(this);
    show_time_label_->setText("Время демонстрации, с");

    url_validator_ = new UrlValidator(this);

    url_input_ = new QLineEdit(this);
    QObject::connect(
        url_input_, &QLineEdit::textChanged, this, &WebsiteForm::onInput
    );
    url_input_->setValidator(url_validator_);

    show_time_input_ = new QSpinBox(this);
    show_time_input_->setMinimum(3);
    show_time_input_->setMaximum(15);
    show_time_input_->setValue(5);

    submit_button_ = new QPushButton(this);
    submit_button_->setText("Сохранить");
    submit_button_->setFixedHeight(50);
    QObject::connect(
        submit_button_, &QPushButton::clicked, this, &WebsiteForm::onSubmit
    );

    QVBoxLayout* layout = new QVBoxLayout();
    layout->addWidget(url_label_);
    layout->addWidget(url_input_);
    layout->addWidget(show_time_label_);
    layout->addWidget(show_time_input_);
```

```
    layout->addWidget(submit_button_);

    QWidget* widget = new QWidget(this);
    widget->setLayout(layout);

    setCentralWidget(widget);

    clear();
}

void WebsiteForm::setFields(WebsiteDisplayInfo info) {
    url_input_->setText(info.url.toString());
    show_time_input_->setValue(info.show_time / 1000);
    submit_button_->setEnabled(url_input_->hasAcceptableInput());
}

void WebsiteForm::clear() {
    row_ = -1;
    url_input_->clear();
    show_time_input_->clear();
    submit_button_->setDisabled(true);
}

void WebsiteForm::onInput() {
    submit_button_->setEnabled(url_input_->hasAcceptableInput());
}

void WebsiteForm::onSubmit(bool) {
    emit submit(WebsiteDisplayInfo{
        QUrl::fromUserInput(url_input_->text()),
        show_time_input_->value() * 1000,
    });
    close();
}

int WebsiteForm::getRow() const { return row_; }

void WebsiteForm::setRow(int row) { row_ = row; }
```

Приложение 3

Исходный код url_validator.hpp

```
#pragma once

#include <QValidator>

class UrlValidator : public QValidator {
    public:
     UrlValidator(QObject* parent = nullptr);

     State validate(QString& input, int& pos) const override;
};
```

Приложение И

Исходный код url_validator.cpp

```
#include "url_validator.hpp"

#include <QUrl>

UrlValidator::UrlValidator(QObject* parent) : QValidator(parent) {}

QValidator::State UrlValidator::validate(QString& input, int& pos) const {
    const QUrl url{input};

    if (!url.isValid()) {
        return QValidator::Intermediate;
    }

    const QList<QString> valid_schemes = {"http", "https"};

    if (!valid_schemes.contains(url.scheme())) {
        return QValidator::Intermediate;
    }

    QString host = url.host();

    if (host.isEmpty()) {
        return QValidator::Intermediate;
    }

    QStringList parts = host.split(".");

    if (parts.size() < 2 || parts.back().isEmpty()) {
        return QValidator::Intermediate;
    }

    return QValidator::Acceptable;
}
```

Приложение К

Исходный код website_list_model.hpp

```
#pragma once

#include "table_list_model.hpp"
#include "website_display_info.hpp"

class WebsiteListModel : public TableListModel<WebsiteDisplayInfo> {
   public:
    WebsiteListModel(QObject* parent = nullptr);

   protected:
    QVariant columnValue(
        const WebsiteDisplayInfo& value,  //
        int column
    ) const override;
};
```

Приложение Л

Исходный код website_list_model.cpp

```
#include "website_list_model.hpp"

WebsiteListModel::WebsiteListModel(QObject* parent) : TableListModel(parent) {}

QVariant WebsiteListModel::columnValue(
    const WebsiteDisplayInfo& value, int column
) const {
    if (column == 0) {
        return value.url.toString();
    } else if (column == 1) {
        return QString("%1c").arg(value.show_time / 1000);
    } else {
        return QVariant();
    }
}
```

Приложение М

Исходный код website_list_window.hpp

```hpp
#pragma once

#include "website_display_info.hpp"
#include "website_form.hpp"
#include "website_list_model.hpp"
#include "website_viewer.hpp"

#include <QMainWindow>
#include <QPushButton>
#include <QTableView>

class WebsiteListWindow : public QMainWindow {
    Q_OBJECT

   public:
    WebsiteListWindow(QWidget* parent = nullptr);

   private:
    void onCreateButtonClicked(bool);

    void onFormSubmit(WebsiteDisplayInfo value);

    void onEditButtonClicked(bool);

    void onDeleteButtonClicked(bool);

    void onViewButtonClicked(bool);

    void openEditForm(QModelIndex index);

    QTableView* table_;
    WebsiteListModel* model_;
    WebsiteViewer* viewer_;
    WebsiteForm* form_;

    QPushButton* create_button_;
```

```
    QPushButton* edit_button_;
    QPushButton* delete_button_;
    QPushButton* view_button_;
};
```

Приложение Н

Исходный код website_list_window.cpp

```
#include "website_list_window.hpp"

#include <QHBoxLayout>
#include <QHeaderView>
#include <QVBoxLayout>

WebsiteListWindow::WebsiteListWindow(QWidget* parent) : QMainWindow(parent) {
    setWindowTitle("Изменение просматриваемых веб-сайтов");

    table_ = new QTableView(this);
    QObject::connect(
        table_,
        &QTableView::doubleClicked,
        this,
        &WebsiteListWindow::openEditForm
    );

    model_ = new WebsiteListModel(this);
    viewer_ = new WebsiteViewer(this);

    form_ = new WebsiteForm(this);
    form_->resize(500, 0);
    QObject::connect(
        form_, &WebsiteForm::submit, this, &WebsiteListWindow::onFormSubmit
    );

    QList<WebsiteDisplayInfo> values;
    values.append(WebsiteDisplayInfo{QUrl("https://google.ru"), 5000});
    values.append(WebsiteDisplayInfo{QUrl("https://ya.ru"), 5000});
    QList<QString> column_names = {"URL", "Время демонстрации"};
    model_->setValues(values);
    model_->setColumnNames(column_names);

    table_->setModel(model_);
    table_->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
    table_->setSelectionBehavior(QTableView::SelectRows);
```

```
table_->setSelectionMode(QTableView::SingleSelection);

create_button_ = new QPushButton(this);
create_button_->setText("Добавить");
create_button_->setFixedHeight(40);
QObject::connect(
    create_button_,
    &QPushButton::clicked,
    this,
    &WebsiteListWindow::onCreateButtonClicked
);

edit_button_ = new QPushButton(this);
edit_button_->setText("Изменить");
edit_button_->setFixedHeight(40);
QObject::connect(
    edit_button_,
    &QPushButton::clicked,
    this,
    &WebsiteListWindow::onEditButtonClicked
);

delete_button_ = new QPushButton(this);
delete_button_->setText("Удалить");
delete_button_->setFixedHeight(40);
QObject::connect(
    delete_button_,
    &QPushButton::clicked,
    this,
    &WebsiteListWindow::onDeleteButtonClicked
);

view_button_ = new QPushButton(this);
view_button_->setText("Просмотр");
view_button_->setFixedHeight(50);
QObject::connect(
    view_button_,
    &QPushButton::clicked,
    this,
    &WebsiteListWindow::onViewButtonClicked
```

```
    );

    QHBoxLayout* button_layout = new QHBoxLayout();

    button_layout->addWidget(create_button_);
    button_layout->addWidget(edit_button_);
    button_layout->addWidget(delete_button_);

    QVBoxLayout* layout = new QVBoxLayout();

    layout->addWidget(table_);
    layout->addLayout(button_layout);
    layout->addWidget(view_button_);

    QWidget* widget = new QWidget(this);
    widget->setLayout(layout);
    setCentralWidget(widget);
}

void WebsiteListWindow::onCreateButtonClicked(bool) {
    form_->clear();
    form_->setWindowTitle("Добавление URL");
    form_->show();
}

void WebsiteListWindow::onFormSubmit(WebsiteDisplayInfo value) {
    int row = form_->getRow();
    if (row == -1) {
        model_->addRow(std::move(value));
    } else {
        model_->setRow(row, std::move(value));
    }
}

void WebsiteListWindow::onEditButtonClicked(bool) {
    QModelIndexList select = table_->selectionModel()->selectedRows();
    if (select.empty()) {
        return;
    }
    openEditForm(select.front());
```

```
}

void WebsiteListWindow::onDeleteButtonClicked(bool) {
    QModelIndexList select = table_->selectionModel()->selectedRows();
    if (select.empty()) {
        return;
    }

    QModelIndex index = select.front();
    model_->removeRow(index.row());
}

void WebsiteListWindow::onViewButtonClicked(bool) {
    viewer_->resize(size());
    viewer_->setWebsiteInfo(model_->values());
    viewer_->show();
}

void WebsiteListWindow::openEditForm(QModelIndex index) {
    int row = index.row();
    form_->clear();
    form_->setFields(model_->getRow(row));
    form_->setRow(row);
    form_->setWindowTitle("Изменение URL");
    form_->show();
}
```

Приложение О

Исходный код website_viewer.hpp

```hpp
#pragma once

#include "website_display_info.hpp"

#include <QLabel>
#include <QMainWindow>
#include <QTimer>
#include <QWebEngineView>

class WebsiteViewer : public QMainWindow {
    Q_OBJECT

   public:
    WebsiteViewer(QWidget* parent = nullptr);

    void setWebsiteInfo(QList<WebsiteDisplayInfo> website_info);

   protected:
    void showEvent(QShowEvent* event) override;

   private:
    void onTimerTimeout();

    void showNextWebsite();

    QLabel* url_label_;
    QWebEngineView* web_view_;
    QList<WebsiteDisplayInfo> website_info_;
    QTimer* timer_;
    int current_website;
    int current_time;
};
```

Приложение П

Исходный код website_viewer.cpp

```
#include "website_viewer.hpp"

#include <QLayout>

WebsiteViewer::WebsiteViewer(QWidget* parent) : QMainWindow(parent) {
    setWindowTitle("Демонстрация веб-сайтов");

    timer_ = new QTimer(this);
    connect(timer_, &QTimer::timeout, this, &WebsiteViewer::onTimerTimeout);

    url_label_ = new QLabel(this);
    QFont font = url_label_->font();
    font.setPixelSize(18);
    url_label_->setFont(font);
    url_label_->setAlignment(Qt::AlignCenter);
    url_label_->setFixedHeight(30);

    web_view_ = new QWebEngineView(this);

    QVBoxLayout* layout = new QVBoxLayout();
    layout->addWidget(url_label_);
    layout->addWidget(web_view_);

    QWidget* window = new QWidget();
    window->setLayout(layout);
    setCentralWidget(window);
}

void WebsiteViewer::onTimerTimeout() { showNextWebsite(); }

void WebsiteViewer::setWebsiteInfo(QList<WebsiteDisplayInfo> website_info) {
    website_info_ = std::move(website_info);
    current_website = 0;
}

void WebsiteViewer::showEvent(QShowEvent* event) {
```

```
    web_view_->show();
    showNextWebsite();
    QMainWindow::showEvent(event);
}

void WebsiteViewer::showNextWebsite() {
    if (current_time > 0) {
        const WebsiteDisplayInfo& info = website_info_[current_website - 1];
        url_label_->setText(QString("URL: %1 Осталось: %2")
                                .arg(info.url.toString())
                                .arg(current_time / 1000));
        current_time -= 1000;
    } else if (current_website < website_info_.size()) {
        const WebsiteDisplayInfo& info = website_info_[current_website];
        current_time = info.show_time;
        web_view_->load(info.url);
        ++current_website;
    } else {
        web_view_->close();
        close();
    }

    if (current_time > 0) {
        timer_->start(1000);
    }
}
```

Приложение Р

Исходный код table_list_model.hpp

```
#pragma once

#include <QAbstractListModel>

template <typename T>
class TableListModel : public QAbstractListModel {
   public:
    TableListModel(QObject* parent = nullptr) : QAbstractListModel(parent) {}

    int rowCount(const QModelIndex&) const override { return values_.count(); }

    int columnCount(const QModelIndex& parent) const override { return 2; }

    QVariant data(const QModelIndex& index, int role) const override {
        const int row = index.row();
        const auto& info = values_.at(row);

        if (role == Qt::DisplayRole) {
            return columnValue(info, index.column());
        } else {
            return QVariant();
        }
    }

    bool isEmpty() const { return values_.isEmpty(); }

    void setColumnNames(QList<QString> column_names) {
        column_names_ = std::move(column_names);
    }

    QVariant headerData(
        int section,  //
        Qt::Orientation orientation,
        int role
    ) const override {
        if (role == Qt::DisplayRole && orientation == Qt::Horizontal &&
```

```
            section < column_names_.size()) {
            return column_names_.at(section);
        }
        return QAbstractListModel::headerData(section, orientation, role);
    }

    const QList<T>& values() const { return values_; }

    void setValues(QList<T> values) {
        int idx = values_.count();
        beginInsertRows(QModelIndex(), 1, idx);
        values_ = std::move(values);
        endInsertRows();
    }

    const T& getRow(int row) { return values_[row]; }

    void addRow(T value) {
        int idx = values_.count();
        beginInsertRows(QModelIndex(), idx, idx);
        values_.append(std::move(value));
        endInsertRows();
    }

    void setRow(int row, T value) { values_[row] = value; }

    void removeRow(int row) {
        beginRemoveRows(QModelIndex(), row, row);
        values_.removeAt(row);
        endRemoveRows();
    }

  protected:
    virtual QVariant columnValue(const T& value, int column) const = 0;

  private:
    QList<T> values_;
    QList<QString> column_names_;
};
```

Приложение С

Исходный код website_display_info.hpp

```
#pragma once

#include <QTime>
#include <QUrl>

struct WebsiteDisplayInfo {
    QUrl url;
    int show_time;
};
```