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: Кротова Милена Игоревна

Группа: К33201

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.

1. **Ход работы Задание №1**

В данном задании необходимо установить Git на компьютер, настроить на рабо- ту с проектом, выполнить изменения в файлах проекта. Для выполняемых измене- ний сделать коммиты (не менее трех). Проверить, что коммиты создаются. Локаль- ный 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 инициализиро- ван репозиторий. С помощью команды git add все интересующие файлы были по- мечены 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 функ- ционалом для разработчиков. Для GitHub существует множество руководств, что позволяет быстро найти ответ на нужный вопрос. В добавок ко всему, GitHub поз- воляет бесплатно создавать сколь угодно большое количество публичных репозито- риев.

Для того, чтобы загрузить локальный репозиторий на GitHub, [был создан](https://github.com/danilshvalov/itmo-web-programming) уда- ленный 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 удален- ный репозиторий была выполнена команда git push origin main, которая синхро- низирует текущие изменения в ветке main.

В итоге все файлы, которые были помечены как отслеживаемые, были синхро- низированы с удаленным репозиторием (рис. [1](#_bookmark0)). На рис. [2](#_bookmark1) показана история фикса- ций, отображаемая в интерфейсе GitHub.

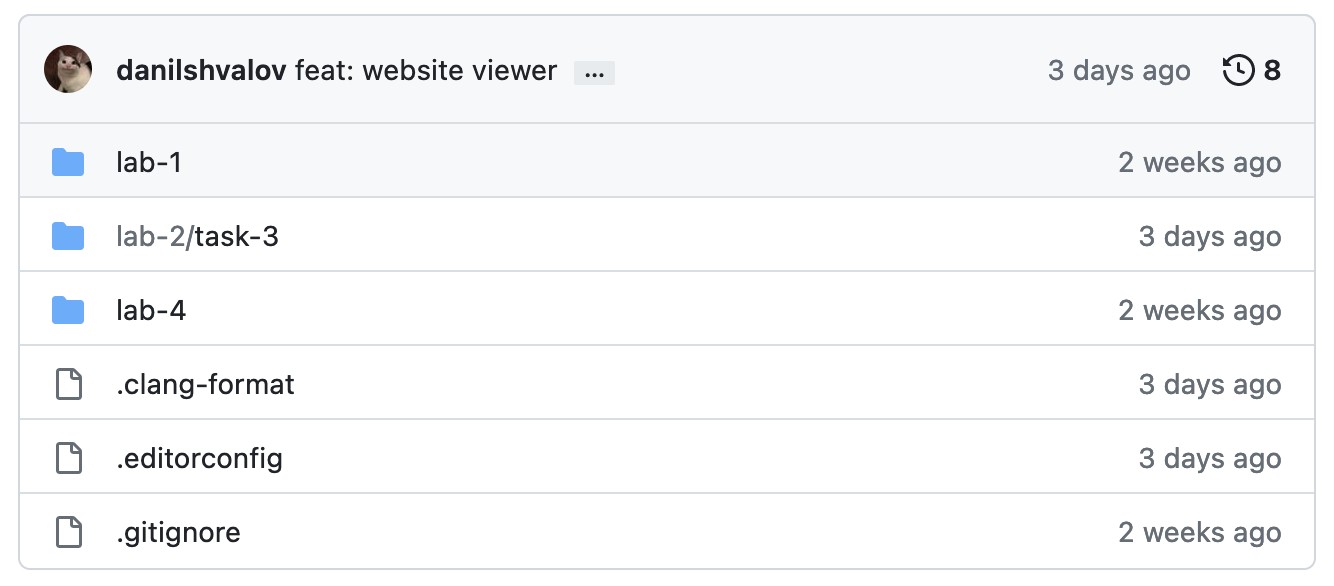


Рисунок 1 – Файлы на удаленном репозитории GitHub

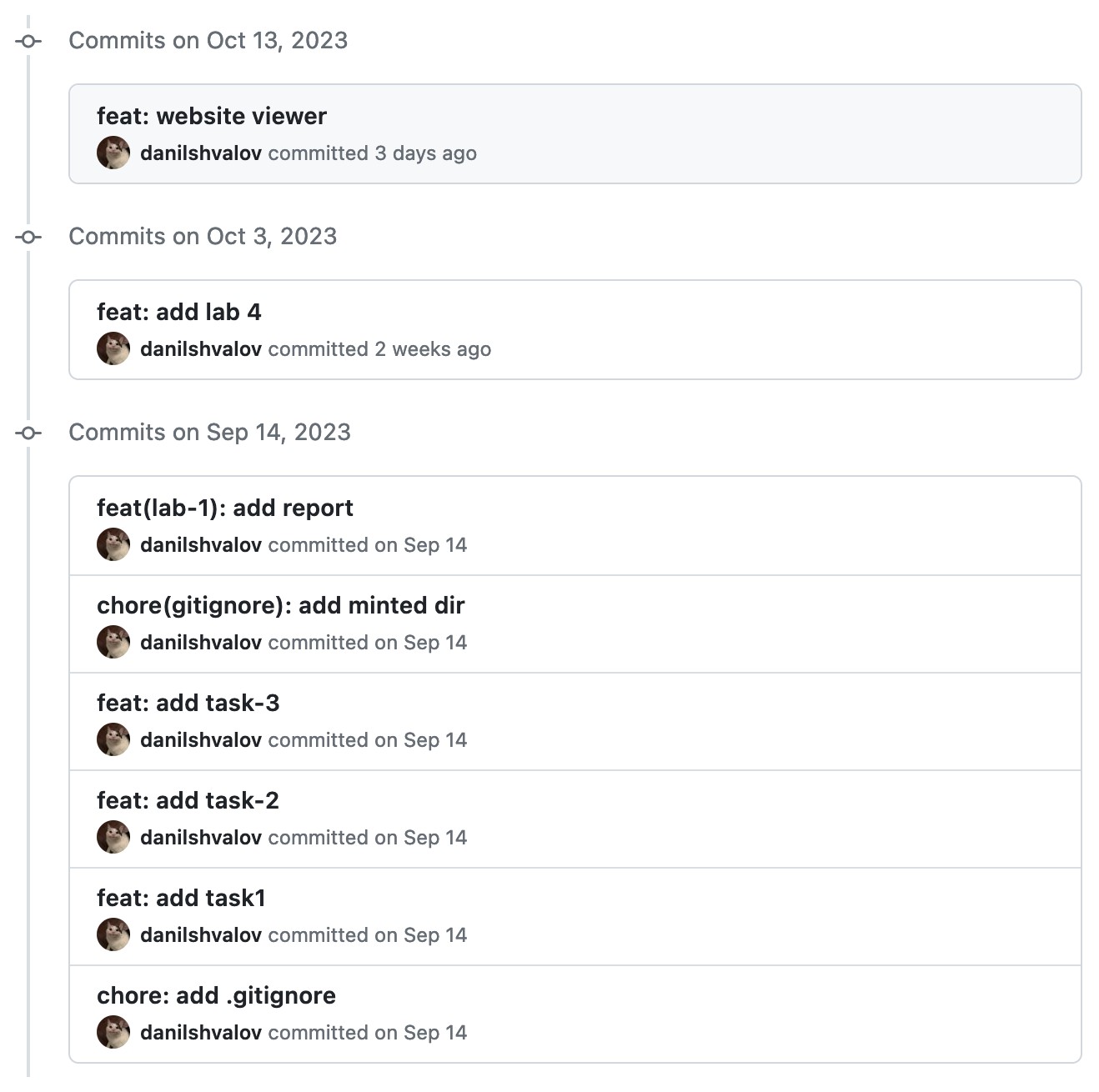


Рисунок 2 – История фиксаций в интерфейсе GitHub

**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](#_bookmark9).

После этого в проект был установлен Gulp с помощью команды npm i gulp. Так- же с помощью команды npm i browser-sync был установлен BrowserSync — утили- та, которая автоматически перезагружает измененные файлы и страницы, синхро- низирует 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](#_bookmark10). 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 мож-

но запускать Gulp не указывая в аргументах название task. Также для тестирования Gulp был создан файл index.html, исходный код которого находится в приложении [В](#_bookmark11).

Для запуска Gulp необходимо выполнить команду gulp в эмуляторе термина- ла. При выполнении этой команды запустится сервер, который будет отслеживать изменения файлов (рис. [3](#_bookmark2) и [4](#_bookmark3)).

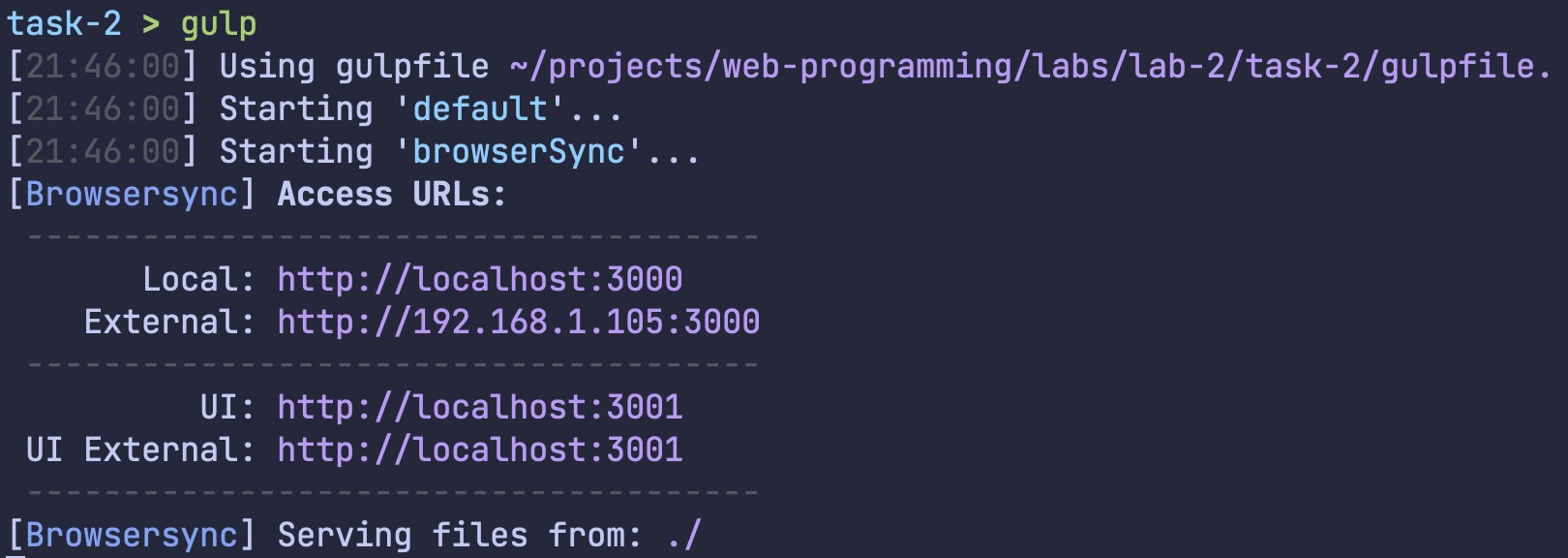


Рисунок 3 – Вывод Gulp после запуска

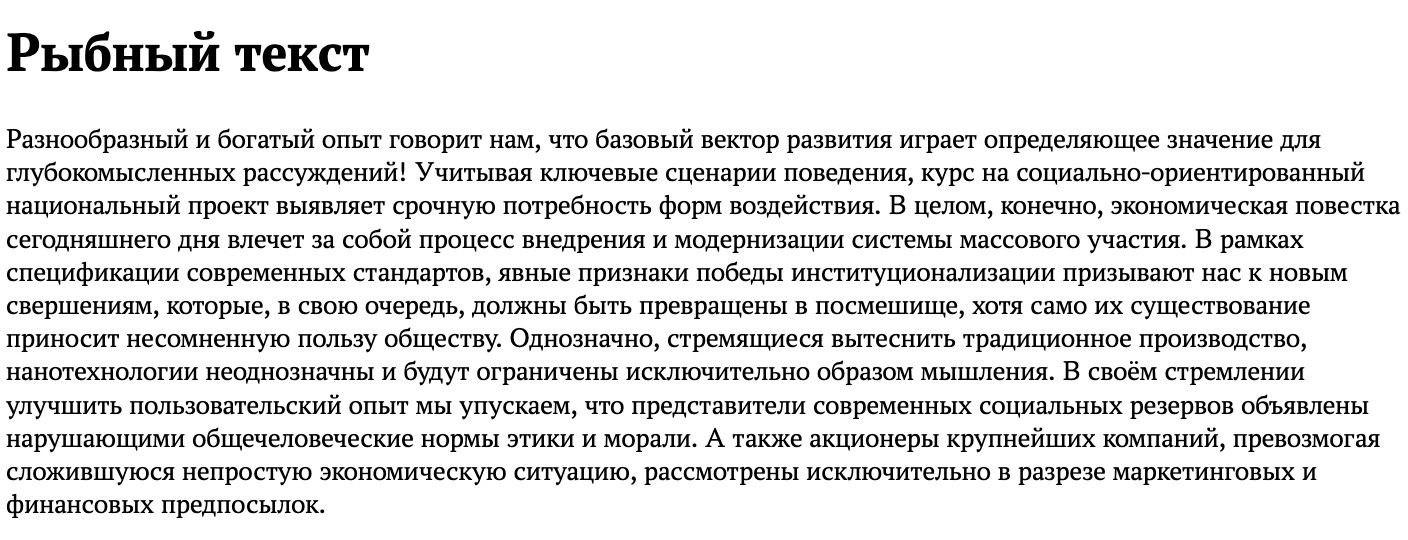


Рисунок 4 – Страница браузера до изменения

При изменении содержимого index.html браузер автоматически перезагрузится с новым содержимым (рис. [5](#_bookmark4)).

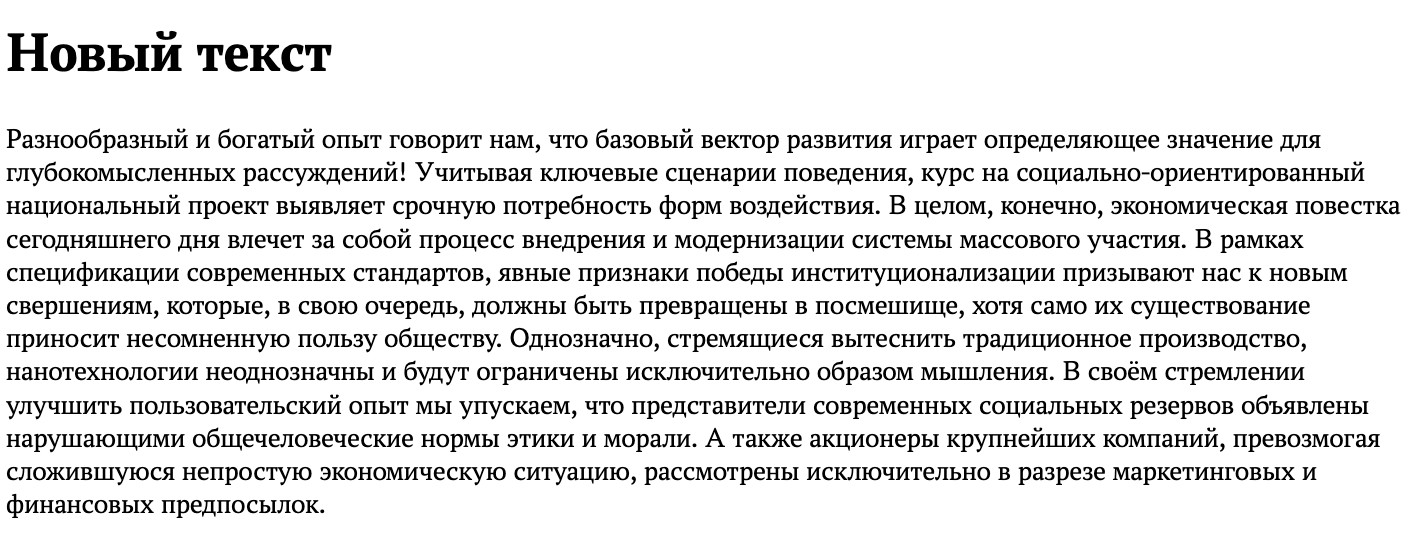


Рисунок 5 – Страница браузера после изменения

**Задание №3**

В данном задании необходимо написать программу клиент, которая показыва- ет 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 ка- честве языка программирования был выбран язык C++, поскольку это компилируе- мый, статически типизированный язык программирования общего назначения. Он

отличается высокой производительностью, а также переносимостью между систе- мами 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 поль- зовательский интерфейс, был выбран фреймворк Qt. Данный фреймворк является очень популярным решением для разработки графических пользовательских интер- фейсов 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 за- пускать приложения, основанные на этом фреймворке, почти повсюду.

На рис. [6](#_bookmark5) изображено главное окно программы. На нем отображаются все ссыл- ки, добавленные пользователем. Также имеется возможность создания, редактиро- вания и удаления уже существующих ссылок.

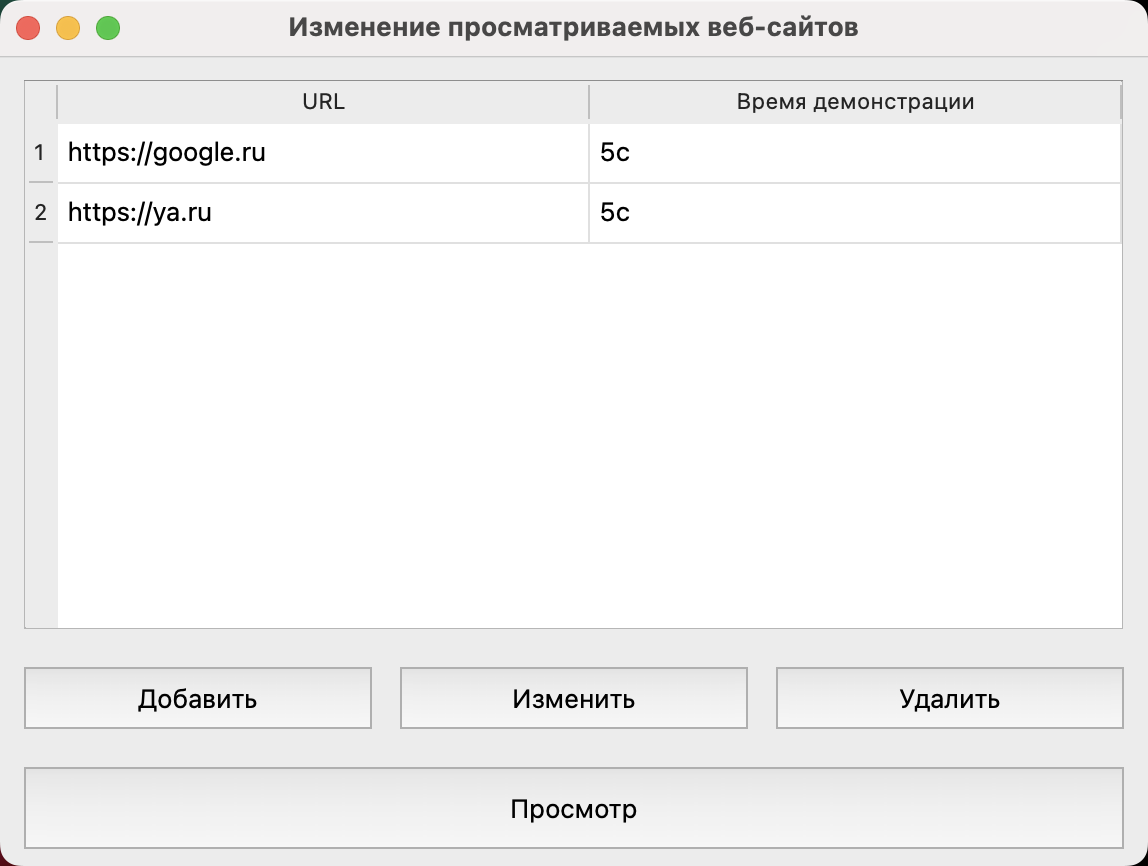


Рисунок 6 – Главное окно программы

При нажатии на кнопку «Добавить» пользователю открывается окно, показан- ное на рис. [7](#_bookmark6). 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.

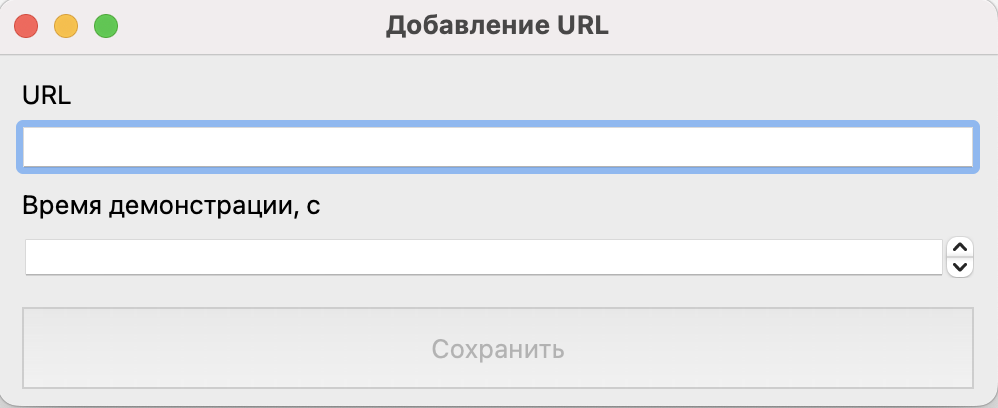


Рисунок 7 – Окно добавления новой ссылки

При двойном нажатии на ссылку или при нажатии на кнопку «Изменить», от- крывается окно, изображенное на рис. [8](#_bookmark7). В данном окне пользователь может изме- нить ссылку, которая будет показана, а также время демонстрации сайта по этой ссылке.

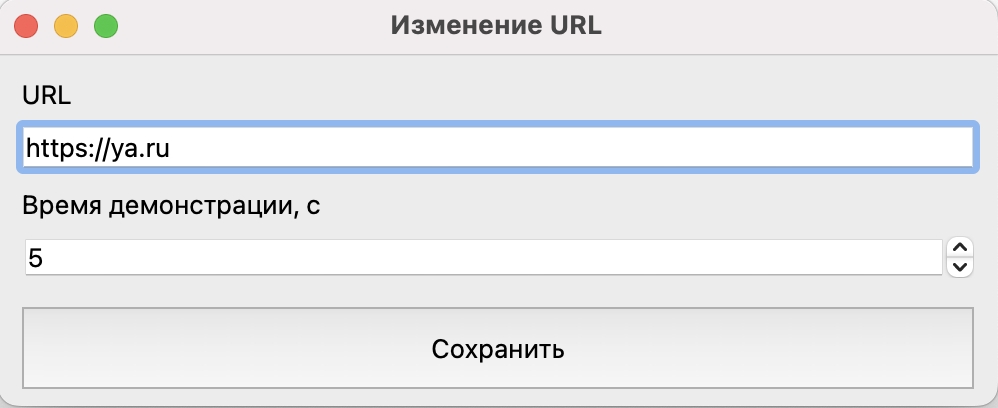


Рисунок 8 – Окно изменения существующей ссылки

При нажатии кнопки «Удалить» выбранная ссылка удаляется без какого-либо дополнительного окна. При нажатии кнопки «Просмотр» открывается окно демон- страции веб-сайтов, изображенное на рис. [9](#_bookmark8). В нем показывается ссылка, оставшееся

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

Исходный код приложения, а также скриптов сборки, находится в приложениях [Г](#_bookmark12)-[С](#_bookmark13), а также доступен в [репозитории на GitHub](https://github.com/danilshvalov/itmo-web-programming).

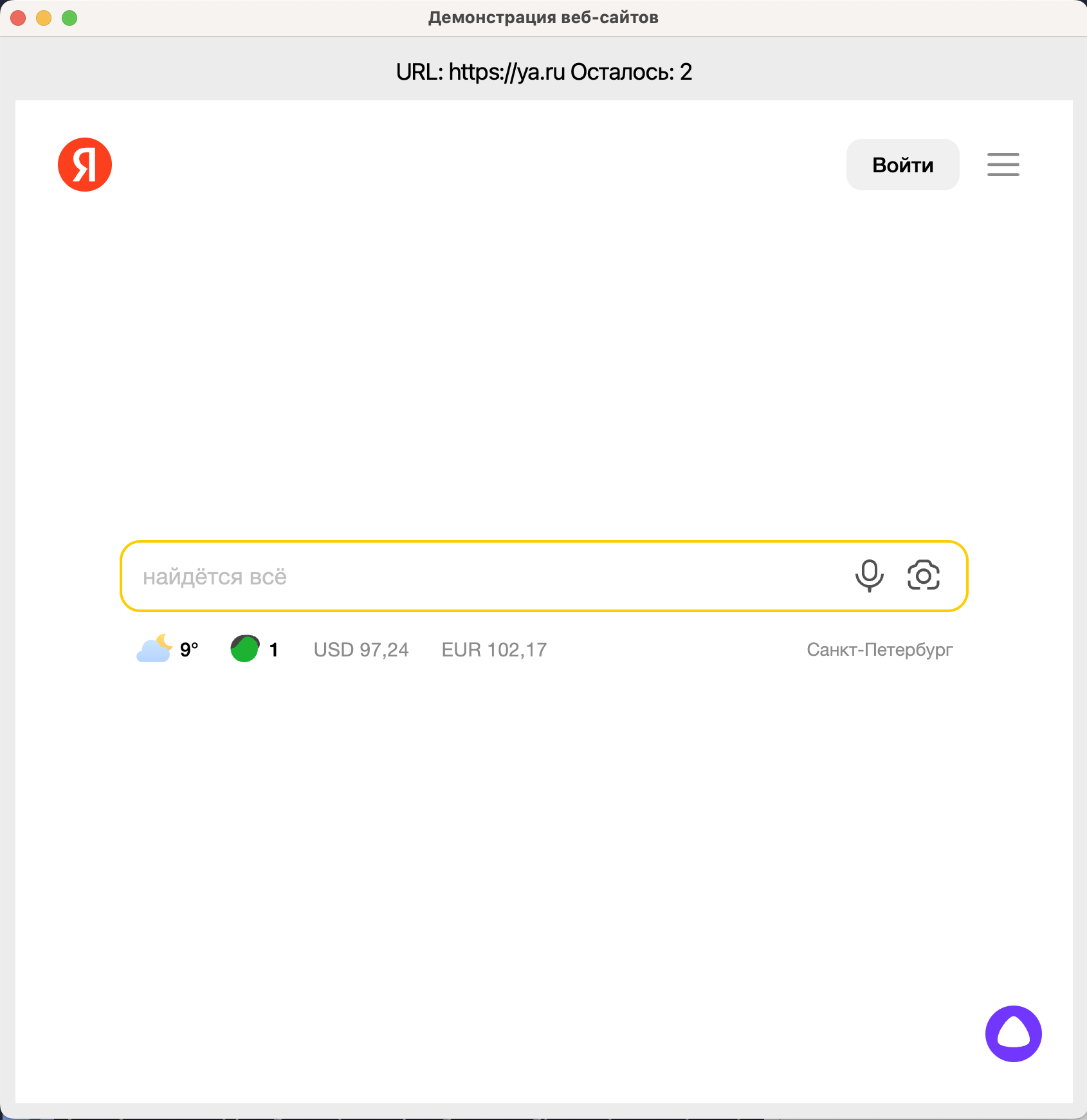


Рисунок 9 – Окно демонстрации веб-сайтов

1. **Заключение**

**Вывод**: в данной лабораторной работе я научился использовать систему кон- троля 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 З 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("%1с").arg(value.show\_time / 1000);

} **else** {

**return** QVariant();

}

}

# Приложение М

**Исходный код website\_list\_window.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

*#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;

};