Частное учреждение образования

«Колледж бизнеса и права»

СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ПО УЧЕБНОМУ ПРЕДМЕТУ ИНСТРУМЕНТАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ИЗУЧЕНИЯ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА И РЕШЕНИЯ БАЗОВЫХ ЗАДАЧ С ВОЗМОЖНОСТЬЮ ПРОВЕРКИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

«Основы алгоритмизации и программирования»

КП Т.295018.401

Руководитель проекта (Дж. А. Миронова)

Учащийся (М. А. Симанович)

2024

Изм.

Лист

№ докум.

Подпись

Датаа

Лист

3

КП Т.295018.401 ПЗ

Разраб.

Симанович М.А.

Провер.

Миронова Дж.А.

Реценз.

Н. Контр.

Утверд.

Создание программного средства по учебному предмету «Инструментальное программное обеспечение» для изучения теоретического материала и решения базовых задач с возможностью проверки

Лит.

Листов

КБиП

50

У

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc170314480)

[1 Описание задачи 5](#_Toc170314481)

[1.1 Анализ предметной области 5](#_Toc170314482)

[1.2 Постановка задачи 6](#_Toc170314483)

[2 Проектирование системы 7](#_Toc170314484)

[2.1 Требования к приложению 7](#_Toc170314485)

[2.2 Проектирование модели 7](#_Toc170314486)

[2.3 Концептуальный прототип 8](#_Toc170314487)

[3 Описание реализации программного средства 11](#_Toc170314488)

[3.1 Инструменты разработки и применяемые технологии 11](#_Toc170314489)

[3.2 Функции: логическая и физическая организация 11](#_Toc170314490)

[3.3 Входные и выходные данные 15](#_Toc170314491)

[3.4 Функциональное тестирование 15](#_Toc170314492)

[3.5 Описание справочной системы 17](#_Toc170314493)

[4 Применение 18](#_Toc170314494)

[4.1 Назначение программного средства 18](#_Toc170314495)

[4.2 Условия применения 18](#_Toc170314496)

[Заключение 19](#_Toc170314497)

[Список использованных источников 20](#_Toc170314498)

[Приложение А 21](#_Toc170314499)

[Приложение Б 48](#_Toc170314500)

Введение

В современном мире трудно представить жизнь без IT-технологий. В этой цифровой эпохе использование программного обеспечения становится неотъемлемой частью нашей реальности. Все ищут способы попасть в IT, выучить язык программирования и вложить свой вклад в развитие технологий. Это инструмент, который облегчает обучение, улучшает процессы обучения и управления информацией, а также способствует развитию профессиональных навыков.

Важность создания программного средства трудно переоценить. Этот инструмент не только обеспечит удобное управление учебным материалом и выполнение практических заданий, но и предоставит пользователю возможность эффективной работы с информацией и обмена опытом с коллегами.

Целью разработки курсового проекта на тему «Создание программного средства по учебному предмету Инструментальное программное обеспечение» является реализация инструмента, который позволит студентам учиться эффективно, проверять свои знания и обмениваться опытом с коллегами, благодаря удобному интерфейсу, возможности выполнять практические задания, формированию отчетов, изучению и пометок в документации.

Актуальность программного средства заключается в том, что оно отвечает современным требованиям к образованию и управлению информацией, предоставляя пользователю инновационные инструменты для обучения и развития профессиональных навыков.

В первом разделе «Описание задачи» основное внимание уделяется выявлению ключевых задач, необходимых для успешной реализации проекта. Это включает в себя выявление основных функциональных требований к программному средству, а также план разработки.

Во втором разделе «Проектирование системы» представлено описание структуры программного средства, включая создание лаконичного интерфейса с использованием меню, кнопочных форм и панелей инструментов, а также организацию иерархии классов и работы с информацией.

В третьем разделе «Реализация программного средства» затронуты технические аспекты создания программного средства, включая выбор используемых технологий, алгоритмы и архитектуру приложения, а также реализацию всех функций и возможностей, описанных в задании.

Четвертый раздел «Применение» посвящен обзору возможностей и сфер применения разработанного программного средства, а также его влиянию на повседневную деятельность пользователей, обучение и управление информацией.

В Заключении проведен анализ результатов разработки, выделены основные достижения и определены перспективы дальнейшего развития проекта.

1 Описание задачи

1.1 Анализ предметной области

Целью разработки курсового проектирования на тему «Создание программного средства по учебному предмету Инструментальное программное обеспечение» является реализация программного средства, которое предоставляет пользователям возможность входа в личный аккаунт или создания нового, просмотра доступных учебных материалов, регистрации в приложении, добавления материалов в избранное и получения краткого описания учебных курсов.

Предметная область языка программирования Python охватывает широкий спектр применений, что делает его одним из самых популярных языков в мире. Python используется во множестве сфер, например В области машинного обучения и искусственного интеллекта Python занимает центральное место, поддерживая такие инструменты как TensorFlow, Keras и PyTorch для разработки и обучения моделей машинного обучения. Python также используется для автоматизации задач и написания скриптов, облегчая системное администрирование и рутинные задачи. В разработке игр Python используется с библиотекой Pygame для создания 2D игр и прототипирования игровых концепций и многое другое.

Компьютерная обработка значительно упрощает процесс доступа к учебным материалам по сравнению с реальным визитом. Она повышает спрос пользователей и их заинтересованность, обеспечивает быстрый доступ к доступным курсам и их стоимости.

Потенциальными пользователями программного средства являются студенты, поскольку учебные курсы востребованы среди молодежи. Программное средство разработано с целью объединения всех необходимых средств для изучения языка программирования.

1.2 Постановка задачи

Задачей проекта является разработка программного средства для просмотра и прохождения учебных курсов. Программа обеспечивает просмотр и ознакомление с курсами, их прохождения, просмотр видеороликов и прохождения практических заданий.

Программное средство будет включать в себя функционал для поиска по названию курса, что позволяет пользователям ориентироваться в большом объеме данных. Также будет возможность сохранения и просмотра пройденных курсов.

На рисунке 1.1 представлен аналог разрабатываемого программного средства «Stepik».

Рисунок 1.1 «Stepik»

Достоинства аналога:

1. Интерфейс: Простой и интуитивно понятный интерфейс облегчает навигацию и использование приложения.

2. Доступность: Возможность регистрации в приложении в любое время и из любого места, используя персональный компьютер

3. История прохождения курсов и отчеты: Хранение истории пройденных курсов, возможность создания отчетов

Недостатки аналога:

1. Зависимость от интернета: Необходимость постоянного подключения к интернету может ограничивать использование приложения в условиях слабого или отсутствующего интернета.

2. Технические проблемы: Возможные ошибки и сбои в работе приложения могут вызвать неудобства и снизить доверие пользователей.

3. Актуальность информации: Необходимость постоянного обновления информации о доступных курсах и их расписании может потребовать значительных усилий от организаторов.

Разрабатываемое программное средство будет включать в себя различные особенности, позволяющие более комфортно взаимодействовать с приложением.

2 Проектирование системы

2.1 Требования к приложению

Безопасность в приложении обеспечена на высоком уровне односторонним шифрованием в хэш и записью в бинарные файлы (.bin). В такие файлы записываются логин, пароль, личные данные пользователя из полей. В приложении реализована авторизация и регистрация, она локальная, потому что приложение поддерживается никакими онлайн серверами и является полностью локальным, что еще больше минимизирует риски взломов.

Интерфейс лаконичен и интуитивно понятен, он не требует от пользователя специальных технических знаний; все элементы управления расположены логично и последовательно. Реализована система вкладок в основной форме, чтобы не нагружать пользователя новыми дополнительными окнами.

Для удобства работы пользователей размеры кнопок и полей ввода, используется современный, читаемый шрифт, Century Gothic размером от 8 до 25 пунктов. Цветовая схема интерфейса темно-синего оттенка, что уменьшает нагрузку на глаза при длительной работе. Применяются сторонние элементы управления из специальных фреймворков, такие как закругленные, анимированные кнопки, выпадающие закругленные списки и закругленные поля ввода, соответствующие принципам удобства использования и красоты.

Минимальные требования к аппаратным и операционным ресурсам:

* OC: Windows 10
* Процессор: Pentium® 4 1.5 GHz / Athlon® XP
* Оперативная память: 2048 MB ОЗУ
* Видеокарта: Встроенная графика или любая дискретная видеокарта.

2.2 Проектирование модели

Для отображения функциональной структуры объектов программного средства необходимо воспользоваться унифицированным языком моделирования UML, а именно диаграмму классов и диаграмму работы алгоритмов.

Диаграмма работы алгоритмов — это структурная диаграмма в языке UML является графическим представлением последовательности шагов выполнения алгоритма. Она помогает визуализировать логику алгоритма и его структуру, делая процесс понятным и легко читаемым. Данная диаграмма полезна при проектировании, отладке и понимании алгоритмов. Они помогают программистам и разработчикам лучше организовать свои мысли, выделить ключевые шаги и легко обнаружить ошибки или улучшить производительность алгоритма. Диаграмма работы алгоритмов представлена в графической части на листе 1.

Диаграмма классов — это структурная диаграмма в языке UML, которая отображает структуру системы, описывая классы, их атрибуты, методы и отношения между ними. Диаграмма классов помогает визуализировать архитектуру программного обеспечения и отношения между объектами. Данная диаграмма помогает разработчикам лучше понять структуру системы, улучшить проектирование программного обеспечения и облегчить коммуникацию между участниками проекта. Они являются важным инструментом для проектирования и документирования программных систем. Диаграмма классов представлена в графической части на листе 2.

2.3 Концептуальный прототип

Концептуальный прототип – это общее видение продукта, его функций, логики взаимодействия с другими компонентами. Цель концептуального прототипа — получить обратную связь от пользователей и заинтересованных сторон на ранних этапах разработки, что позволяет уточнить требования и улучшить дизайн до начала полноценной разработки.

Окно авторизации включает в себя список кнопок, которые выполняют различные действия. Графическое представление прототипа окна авторизации представлено на рисунке 2.1.

Рисунок 2.1 – прототип окна авторизации

Кнопка регистрация оставляет пользователя на этой форме, добавляет кнопку зарегистрироваться и textBox для подтверждения пароля. Прототип представлен на рисунке 2.2.

Рисунок 2.2 – окно регистрации

Кнопка вход перебрасывает пользователя на рабочее окно приложения. Прототип рабочего окна приложения представлен на рисунке 2.3.

Рисунок 2.3 – прототип рабочего окна приложения

После нажатия кнопки регистрации идет перенаправление на окно заполнения профиля. Где он должен подтвердить свой email и указать личные данные. Прототип предоставлен на рисунке 2.4

Рисунок 2.4 – прототип окна заполнения профиля

3 Описание реализации программного средства

3.1 Инструменты разработки и применяемые технологии

Разработка приложения на С++ с использованием фреймворков Guna2UI, SiticoneDesktop и BunifuFramework в среде Visual Studio представляет собой процесс, направленный на создание высокопроизводительного клиентского приложения с современным и эстетичным пользовательским интерфейсом.

Язык программирования C++ выбран из-за его высокой производительности и возможности эффективного управления ресурсами, что особенно важно для приложений, взаимодействующих с данными из интернета, такими как данные о криптовалютах. Visual Studio, в качестве интегрированной среды разработки, обеспечивает удобную платформу для создания, отладки и развертывания приложений на C++. Она интегрирует в себя все необходимые инструменты и ресурсы для эффективной работы над проектами.

Фреймворки Guna2UI, SiticoneDesktop и BunifuFramework предоставляют разнообразные графические компоненты и элементы управления, которые позволяют создавать современные пользовательские интерфейсы. Они предоставляют не только стандартные элементы WinForms, но и расширенные возможности для стилизации и анимации интерфейса, что значительно повышает его привлекательность и удобство использования для конечного пользователя.

Такой подход позволяет разработать полноценное обучающее приложение, способное эффективно обрабатывать код, предоставляя пользователю удобный и современный интерфейс для взаимодействия с документацией, задачами и другими данными.

3.2 Функции: логическая и физическая организация

Реализация системы регистрации. Эта функция выполняется на стадии регистрации логина и пароля, она проверяет поля на корректность, в случае если какое-либо поле некорректно, она завершает свою работу. В ином случае функция вызывает другую, которая записывает в файл данные.

Void auth::buttonRegister\_Click(Object^ sender, EventArgs^ e) {

// Проверка на пустые поля

if (textBoxLogin1->Text == "") {

textBoxLogin1->BorderColor = Color::Red;

MessageWarning->Show("Введите логин");

return;

}

if (textBoxPassword1->Text == "" || textBoxPassword\_1->Text == "") {

textBoxPassword1->BorderColor = Color::Red;

textBoxPassword\_1->BorderColor = Color::Red;

MessageWarning->Show("Введите пароль");

return;

}

// Проверка на совпадение паролей

if (textBoxPassword1->Text != textBoxPassword\_1->Text) {

textBoxPassword1->BorderColor = Color::Red;

textBoxPassword\_1->BorderColor = Color::Red;

MessageError->Show("Пароли не совпадают", "Ошибка");

return;

}

// Проверка на соглашение

if (checkBoxTerms->Checked == false) {

MessageError->Caption = "Предупреждение";

MessageError->Text = "Для регистрации вам нужно прочесть условия использования";

MessageError->Show();

return;

}

// Проверка на существование уже такого пользователя

if (Directory::Exists(textBoxLogin1->Text)) {

MessageError->Caption = "Предупреждение";

MessageError->Text = "Пользователь под таким логином уже существует";

MessageError->Show();

return;

}

// Регистрация в случае если все условия соблюдены

panelMain->Visible = false;

Register();

}

Реализация функции Register(). Эта функция сохраняет логин, пароль пользователя.

Void auth::Register() {

String^ cliLogin = textBoxLogin1->Text;

String^ cliPassword = textBoxPassword1->Text;

USER = cliLogin;

labelWelcome->Text += cliLogin;

SetCenter(pageWelcome, labelWelcome, 1);

std::string Login = msclr::interop::marshal\_as<std::string>(cliLogin);

std::string Password = msclr::interop::marshal\_as<std::string>(cliPassword);

String^ folderPath = cliLogin;

Directory::CreateDirectory(folderPath);

String^ filePath = Path::Combine(folderPath, "data.bin");

std::hash<std::string> hasher;

size\_t hashPassword = hasher(Password);

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Create, FileAccess::Write);

BinaryWriter^ binaryWriter = gcnew BinaryWriter(fileStream);

binaryWriter->Write(hashPassword);

binaryWriter->Close();

fileStream->Close();

animation\_mode = 1;

sec = 4;

timerTransition->Start();

}

Реализация функции входа в приложение:

Обработчик события кнопки «Войти» проверяет корректность введенных данных в поля, активировал ли пользователь функцию запоминания данных при последующих входах, и вызывает функцию входа типа Boolean, которая проверяет данные сохранённые в файлы и введенные пользователем. В случае сходства данных функция возвращает true иначе false.

Если функция возвращает true, обработчик активирует анимацию перехода на основную форму. Приветствие показывается если пользователь включил это в настройках.

Void auth::buttonComeIn\_Click(Object^ sender, EventArgs^ e) {

if (textBoxLogin->Text == "") {

textBoxLogin->BorderColor = Color::Red;

MessageWarning->Show("Введите логин");

return;

}

if (textBoxPassword->Text == "") {

textBoxPassword->BorderColor = Color::Red;

MessageWarning->Show("Введите пароль");

return;

}

if (ComeIn()) {

if (SwitchRemember->Checked == true) {

CreateLogFile();

}

if (greeting) {

animation\_mode = 4;

sec = 4;

}

else {

animation\_mode = 5;

sec = 1;

}

labelWelcome->Text += USER;

SetCenter(pageWelcome, labelWelcome, 1);

timerTransition->Start();

}

}

Boolean auth::ComeIn() {

String^ cliLogin = textBoxLogin->Text;

String^ cliPassword = textBoxPassword->Text;

std::string Login = msclr::interop::marshal\_as<std::string>(cliLogin);

std::string Password = msclr::interop::marshal\_as<std::string>(cliPassword);

String^ folderPath = cliLogin;

String^ filePath = Path::Combine(folderPath, "data.bin");

if (!Directory::Exists(folderPath)) {

MessageError->Caption = "Ошибка";

MessageError->Text = "Пользователя с указанным логином не существует";

MessageError->Show();

return false;

}

std::hash<std::string> hasher;

size\_t hashPasswordInput = hasher(Password);

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Open, FileAccess::Read);

BinaryReader^ binaryReader = gcnew BinaryReader(fileStream);

size\_t hashPasswordFile = binaryReader->ReadUInt64();

binaryReader->Close();

fileStream->Close();

if (hashPasswordInput == hashPasswordFile) {

USER = cliLogin;

return true; // Пароль совпадает

}

else {

MessageError->Caption = "Предупреждение";

MessageError->Text = "Введен неверный пароль";

MessageError->Show();

return false; // Пароль не совпадает

}

}

Текст остальных программных модулей представлен в приложении А.

3.3 Входные и выходные данные

Входные данные:

1. Логин и пароль: Строка, содержащая логин из букв и (или) цифр.
2. Имя и Фамилия: Строка, содержащая имя пользователя из букв и (или) цифр.
3. Дата рождения: Выбранные элементы из массива.
4. Электронная почта: Строка, являющаяся адресом, которая имеет ключевые символы.
5. Исходный код на Python: Строка, содержащая код задачи на языке Python.
6. Аватар пользователя: Изображение, имеющее jpg или png формат.

Выходные данные:

1. Логин: Папка, имеющая наименование равное логину пользователя
2. Файл с паролем: Хешированная строка, записанная в бинарном файле.
3. Файл с датой рождения: Дата, содержащая информацию о дате рождения пользователя, записанная в бинарном виде.
4. Файл с именем: Строка, содержащая имя пользователя.
5. Файл с полом: Строка, содержащая пол пользователя.
6. ­Файл с фамилией: Строка, содержащая фамилию пользователя, если он её указывал.
7. Файл с прогрессом: Число, преобразованное в строку и записанная в бинарный файл с помощью специального ключа шифрования.
8. Файл с уровнем: Строка, содержащая информацию о уровне пользователя.
9. Файл с состоянием задач: Словарь, содержащий информацию о состояния решения каждой задачи.
10. Тесты для задачи: Скрипт, содержащий Unit-тесты написанные на Python, который проверяет исходный код пользователя на Python.
11. Файл отчета скрипта: Строка, сообщающая программе результаты тестов.
12. Файл решения: Отчёт, содержащий код решения задачи пользователя.

3.4 Функциональное тестирование

Функциональное тестирование – это вид тестирования, который проверяет соответствие функциональности продукта тому, как он был задуман. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тест-кейсы для основных функций представлены в таблице 3.1.

Таблица 3.1 – Функциональное тестирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №№ | Модуль / Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
|  | Авторизация пользователей | 1. Ввести существующие и верные логин и пароль. 2. Нажать кнопку «Войти» | Открытие главного окна на странице «Главная» | Результат представлен на рисунке Б.1, Б.2 и Б.3 |
|  | Функция «Запомнить меня» | 1. Ввести существующие и верные логин и пароль. 2. Включение функции нажатием на кнопку «Запомнить меня». 3. Нажать кнопку «Войти» | Открытие главного окна на странице «Главная» при запуске файла | Результат представлен на рисунке Б.4 и Б.5 |
|  | Регистрация пользователей | 1. Ввести корректные логин, пароль и пароль повторно. 2. Прочитать и принять условия пользования нажав «Я принимаю условия использования» 3. Нажать на кнопку «Регистрация» | Перенаправление на начальное окно заполнения профиля | Результат представлен на рисунке Б.6, Б.7, Б.8, Б.9, Б.10, Б.11 и Б.12 |
|  | Заполнение начального профиля | 1. Ввести корректные личные данные 2. Ввести Email 3. Нажать на кнопку «Отправить письмо» 4. Ввести полученный код в поле 5. Загрузить аватар 6. Нажать на кнопку «Продолжить» | Перенаправление на страницу настроек | Результат представлен на рисунке Б.13, Б.14 и Б.15 |
|  | Первоначальная настройка | 1. Изменить нужные параметры 2. Нажать «Продолжить» | Перезапуск программного средства | Результат представлен на рисунке Б.16 |
|  | Просмотр личной информации | 1. Выбрать страницу «Профиль». 2. Ознакомиться с информацией | Предоставленные личные данные пользователю | Результат представлен на рисунке |
|  | Редактирование личной информации | 1. Выбрать страницу «Профиль» 2. Нажать на кнопку «Редактировать» 3. Редактировать разрешенные поля | Измененные личные данные, сохранение в файлы | Результат представлен на рисунке |
|  | Реализация теоретического материала по курсам, с возможностью создания пометок в предоставленном материале | 1. Выбрать страницу «Книга» 2. Ознакомиться с материалом 3. Выделить текст 4. Пометить нужным способом 5. Очистить ненужные пометки | Изученный материал, сохраненные пометки в файлы | Результат представлен на рисунке |
|  | Реализация возможности прохождения практических заданий в виде решения задач при помощи интерпретатора Python и предоставление результата при о пройденных заданиях. | 1. Выбрать страницу «Задания» 2. Выбрать задание 3. Ознакомиться с поставленной задачей 4. Написать код в поле 5. Нажать «Проверить» 6. Нажать «Сохранить» | Пройденная задача, предоставленный результат, сохранение данных в файлы | Результат представлен на рисунке |
|  | Формирование отчетов прохождения практических заданий за выбранный промежуток времени. | 1. Выбрать страницу «Задания» 2. Выбрать промежуток времени, нажав «Период» 3. Нажать «Отчет» | Предоставленный отчет, возможность экспорта | Результат представлен на рисунке |
|  | Экспортирование отчетов | 1. Нажать «Отчет» 2. Подтвердить действие, нажав «Да» 3. Выбрать путь к файлу 4. Нажать «Сохранить» | Экспортированный отчет в выбранную директорию | Результат представлен на рисунке |
|  | Организация лаконичного интерфейса программного средства с помощью создания меню, кнопочных форм, панелей инструментов | * Выбор страницы * Просмотр анимации * Настройка внешнего вида | Удобный переход по страницам, анимированные элементы | Результат представлен на рисунке |
|  | Справочная система | 1. Нажать на иконку в виде книги с вопросом 2. Ознакомиться со справочной системой | Открытая справочная система | Результат представлен на рисунке |
|  | Реализация сохранения данных в файлы формата \*.docx, \*.txt, \*.bin, \*.rtf и чтение этих данных из них | * Изменить профиль * Изменить настройки * Решить задачу * Провести время в программном средстве | Сохранение данных в файлы | Результат представлен на рисунке |

3.5 Описание справочной системы

Средством разработки для создания справочной системы является программное средство Dr.Explain, которое помогает создавать файлы справки, документацию и руководства пользователя [7].

Справочная система состоит из начальной страницы, оглавления, 9-ти разделов и 3-х подразделов.

Раздел 1 «Запуск приложения» Предоставляет описание процесса запуска приложения в системе.

Подраздел 1.1 «Шаги по установке Python» является инструкцией по установке Python, для работы скрипта программного средства.

Раздел 2 «Авторизация» Является подробным обзором страницы входа.

Раздел 3 «Регистрация» Предоставляет инструкцию по регистрации.

Подраздел 3.1 «Заполнение профиля» Предоставляет подробную инструкцию по начальному заполнению профиля и личной информации. Подраздел 3.2 «Начальная настройка» Предоставляет инструкцию по ранней настройке внешнего вида приложения.

Раздел 4 «Главная страница» Предоставляет информацию об элементах на главной странице программного средства.

Раздел 5 «Книга» Предоставляет инструкцию по использованию функций книги.

Раздел 6 «Задания» Предоставляет подробную информацию о странице задач и о самих задачах.

Подраздел 6.1 «Отчеты» Предоставляет инструкция об отчетах

Раздел 7 «Профиль» Предоставляет пользователю информацию о его личных данных и профиле, а также возможности редактирования этих данных.

Раздел 8 «Статистика» Обозревает функции страницы статистики.

Раздел 9 «Страница настроек» Явно показывает функционал и советы по настройке.

Справочная система вызывается c помощью нажатия клавиши F1 на клавиатуре.

4 Применение

4.1 Назначение программного средства

Программное средство «PythonWave» предназначено для изучения языка программирования Python. Пользователи могут использовать это программное средство для обучения, создание отчётов своей продуктивности и получение базы Python. Так же познакомившись с этим языком, пользователь может понять симпатизирует ли ему этот язык в целом.

Ограничение, накладываемое на представленное программное средство, включает следующее: необходимость наличия операционной системы Windows.

4.2 Условия применения

Для функционирования программы требуется операционная система Windows, интерпретатор Python (рекомендуемые версии 3.12.3 или 3.12.4). Процесс запуска приложения заключается в перемещения содержимого папки «PythonWave» в выбранную директорию на компьютере и запуске файла «PythonWave.exe».

Заключение

Целью курсового проектирования было создание программного средства с целью обучения пользователя. Оно позволяет выполнять практические задачи, изучать теоретический материал, создавать отчеты, настраивать приложение. Была поставлена задача реализовать данное программное средство в среде разработки Visual Studio, на языке программирования C++ с использованием фреймворка CLI .NET. В процессе разработки серьезных трудностей не возникало.

В процессе разработки курсового проектирования были решены следующие задачи: Организация работы с файлами и xml конфигуратором; разработка классов для отслеживания прогресса и решенных задач; специальный класс анимации для управления прозрачности формы; обработка состояний наведения на элементы формы; обеспечение безопасности путем шифрования специальными ключами и хэшем, а также использованием электронной почты; разработка unit-тестов на Python для проверки кода.

Достоинством программного средства является простое взаимодействие пользователя с приложением. Удобный и лаконичный интерфейс с анимациями делает работу с программой максимально эффективной и приятной. Пользователю не нужно тратить много времени на изучение сложных инструкций или освоение новых навыков — все действия выполняются легко и быстро. Программа также обеспечивает высокий уровень интерактивности, что способствует лучшему усвоению материала.

Недостатком является то, что приложение является локальным. Пользователь не сможет войти в свой аккаунт на других устройствах, где нет сохраненных файлов. Это ограничивает мобильность и гибкость использования программного средства.

Улучшение данного программного средства возможно путем полного перехода на язык программирования C#, так как на нем корректно работают сторонние фреймворки. Подключение базы данных SQL для сохранения данных пользователя онлайн позволит создать систему учета прогресса, доступную с любого устройства. Также стоит рассмотреть доработку и пополнение задач для повышения их разнообразия и уровня сложности, что сделает обучение более комплексным и полезным.

Дополнительно можно рассмотреть внедрение возможностей для совместного обучения, таких как обмен задачами между пользователями и поддержка учебных групп, что позволит создать более интерактивное и социальное учебное окружение. Интеграция с популярными облачными сервисами для хранения данных также может повысить удобство использования приложения.

Список использованных источников

1. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2013. – 29 c.
2. Блок-схема [Электронный ресурс]. – Wikipedia, 2024. – Режим доступа : https://ru.wikipedia.org/wiki/Блок-схема. – Дата доступа : 15.04.2024.
3. Диаграмма классов [Электронный ресурс]. – Wikipedia, 2024. – Режим доступа : https://ru.wikipedia.org/wiki/Диаграмма\_классов. – Дата доступа : 17.04.2024.
4. C++ [Электронный ресурс]. – Wikipedia, 2024. - Режим доступа : https://ru.wikipedia.org/wiki/C++. – Дата доступа : 11.05.2024.
5. Visual Studio 2022 [Электронный ресурс]. – Microsoft, 2024. – Режим доступа : https://visualstudio.microsoft.com/ru/vs. – Дата доступа : 13.05.2024.
6. Windows Forms [Электронный ресурс]. – Wikipedia, 2024. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/. Дата доступа : 15.05.2024.
7. Guna Framework [Электронный ресурс]. – gunaui.com, 2024. – Режим доступа: <https://gunaui.com/>. – Дата доступа : 17.05.2024.
8. Bunifu Framework [Электронный ресурс]. – bunifuframework.com, 2024. – Режим доступа: https://bunifuframework.com/. – Дата доступа : 20.05.2024.
9. Dr.Explain [Электронный ресурс]. – drexplain.ru, 2024. – Режим доступа: https://www.drexplain.ru/. – Дата доступа : 25.05.2024.

Приложение А

**(обязательное)**

**Текст программных модулей**

«main.cpp»

#include "auth-header.h"

#include "profile-header.h"

#include "main-header.h"

using namespace PythonWave;

[STAThread]

int main(array<System::String^>^ args) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

auth form;

form.StartPosition = FormStartPosition::Manual;

form.Location = Point((Screen::PrimaryScreen->Bounds.Width - form.Width) / 2,

(Screen::PrimaryScreen->Bounds.Height - form.Height) / 2);

Application::Run(% form);

}

void authToProfile(String^ construct) {

profile^ profileForm = gcnew profile(construct);

profileForm->StartPosition = FormStartPosition::Manual;

profileForm->Location = Point((Screen::PrimaryScreen->Bounds.Width - profileForm->Width) / 2,

(Screen::PrimaryScreen->Bounds.Height - profileForm->Height) / 2);

for each (Form ^ form in Application::OpenForms)

{

if (form->GetType() == auth::typeid)

{

form->Hide();

break;

}

}

profileForm->Show();

}

void authToMain(String^ construct) {

mainForm^ mainF = gcnew mainForm(construct);

mainF->StartPosition = FormStartPosition::Manual;

mainF->Location = Point((Screen::PrimaryScreen->Bounds.Width - mainF->Width) / 2,

(Screen::PrimaryScreen->Bounds.Height - mainF->Height) / 2);

for each (Form ^ form in Application::OpenForms)

{

if (form->GetType() == auth::typeid)

{

form->Hide();

break;

}

}

mainF->Show();

}

void mainToAuth() {

if (Directory::Exists("logs")) {

Directory::Delete("logs", true);

}

else {

MessageBox::Show("Error");

}

auth^ form = gcnew auth();

form->StartPosition = FormStartPosition::Manual;

form->Location = Point((Screen::PrimaryScreen->Bounds.Width - form->Width) / 2,

(Screen::PrimaryScreen->Bounds.Height - form->Height) / 2);

for each (Form ^ form in Application::OpenForms)

{

if (form->GetType() == auth::typeid)

{

form->Hide();

break;

}

}

form->Show();

}

void SetCenter(Control^ background, Control^ control, int mode = 0) {

int panelWidth = background->Width;

int panelHeight = background->Height;

int controlWidth = control->Width;

int controlHeight = control->Height;

int newX = (panelWidth - controlWidth) / 2;

int newY = (panelHeight - controlHeight) / 2;

switch (mode)

{

case 0:

MessageBox::Show("background: " + background->Name + "\nelement: " + control->Name, "Error: SetCenter mode = 0");

break;

case 1:

control->Location = Point(newX, control->Location.Y);

break;

case 2:

control->Location = Point(control->Location.X, newY);

break;

case 3:

control->Location = Point(newX, newY);

break;

default:

MessageBox::Show("Ошибка в функции SetCenter", "Неверное значение mode");

break;

}

}

String^ GetFolderCreationDate(String^ folderPath) {

pin\_ptr<const wchar\_t> wch = PtrToStringChars(folderPath);

const std::wstring& folderPathStr = wch;

WIN32\_FILE\_ATTRIBUTE\_DATA fileInfo;

if (GetFileAttributesEx(folderPathStr.c\_str(), GetFileExInfoStandard, &fileInfo)) {

FILETIME creationTime = fileInfo.ftCreationTime;

SYSTEMTIME systemTime;

FileTimeToSystemTime(&creationTime, &systemTime);

String^ creationDateString = String::Format("{0}/{1}/{2}", systemTime.wDay, systemTime.wMonth, systemTime.wYear);

return creationDateString;

}

else {

return "Failed to get folder creation date.";

}

}

«auth-header.h»

#pragma once

#include "auth.h"

#include "ClassFade.h"

#include "config.h"

#include <Windows.h>

#include <iostream>

#include <msclr\marshal\_cppstd.h>

using namespace System;

using namespace PythonWave;

using namespace System::IO;

int sec = 4;

int animation\_mode = 1; // 1,2,3 - Цепочка анимации регистрации, 4 - Переход на main с приветствием

bool greeting = true;

void authToProfile(String^ construct); // Функция для перехода определена в general.h

void authToMain(String^ construct); // Функция для перехода определена в general.h

void SetCenter(Control^ back, Control^ control, int mode); // Функция чтобы установил элемент по середине

String^ auth::ReadLogFile() {

String^ content = "";

try {

String^ folderName = "logs";

String^ filePath = Path::Combine(folderName, "logs.bin");

if (File::Exists(filePath)) {

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Open, FileAccess::Read);

BinaryReader^ binaryReader = gcnew BinaryReader(fileStream);

content = binaryReader->ReadString();

binaryReader->Close();

fileStream->Close();

}

}

catch (Exception^ e) {

MessageError->Show(e->Message);

}

return content;

}

Void auth::CreateLogFile() {

try {

String^ folderName = "logs";

String^ filePath = Path::Combine(folderName, "logs.bin");

Directory::CreateDirectory(folderName);

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Create, FileAccess::Write);

BinaryWriter^ binaryWriter = gcnew BinaryWriter(fileStream);

binaryWriter->Write(USER);

binaryWriter->Close();

fileStream->Close();

}

catch (Exception^ e) {

MessageError->Show(e->Message);

}

}

Void auth::LastEnter() {

if (greeting) {

labelWelcome->Text += USER;

SetCenter(pageWelcome, labelWelcome, 1);

pagesTransition->SelectTab(pageWelcome);

animation\_mode = 4;

sec = 2;

timerTransition->Start();

}

else {

ClassFade^ fadeEffect = gcnew ClassFade(this);

fadeEffect->SetAnimation("hide");

fadeEffect = nullptr;

authToMain(USER);

}

}

Void auth::loadConfig() {

if (File::Exists("config.xml")) {

Config^ config = config->LoadConfig();

borderlessForm->BorderRadius = config->borderForm;

borderlessForm->HasFormShadow = config->hasFormShadow;

buttonComeIn->BorderRadius = config->borderBtn;

buttonRegister->BorderRadius = config->borderBtn;

greeting = config->greeting;

dragControl1->TransparentWhileDrag = config->dragTransparent;

dragControl2->TransparentWhileDrag = config->dragTransparent;

dragControl3->TransparentWhileDrag = config->dragTransparent;

dragControlMain->TransparentWhileDrag = config->dragTransparent;

dragControlMain2->TransparentWhileDrag = config->dragTransparent;

}

}

Void auth::auth\_Load(System::Object^ sender, System::EventArgs^ e) {

if (File::Exists("config.xml")) {

loadConfig();

}

if (Directory::Exists("logs")) {

USER = ReadLogFile();

LastEnter();

}

else {

ClassFade^ fadeEffect = gcnew ClassFade(this);

fadeEffect->SetAnimation("in");

fadeEffect = nullptr;

}

SetCenter(panelMain, labelMain, 1);

SetCenter(pageWelcome, labelWelcome, 1);

}

Void auth::buttonClose\_Click(System::Object^ sender, System::EventArgs^ e) {

ClassFade^ fadeEffect = gcnew ClassFade(this);

fadeEffect->SetAnimation("close");

fadeEffect = nullptr;

}

Void auth::buttonMinimize\_Click(System::Object^ sender, System::EventArgs^ e) {

ClassFade^ fadeEffect = gcnew ClassFade(this);

fadeEffect->SetAnimation("minimize");

fadeEffect = nullptr;

}

Void auth::buttonQuestion\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageOption->Show();

}

Void auth::buttonShowAuth\_Click(System::Object^ sender, System::EventArgs^ e) {

pagesAuthReg->SelectTab(pageAuth);

this->Text = "PythonWave: Авторизация";

labelMain->Text = this->Text;

SetCenter(panelMain, labelMain, 1);

}

Void auth::buttonShowReg\_Click(System::Object^ sender, System::EventArgs^ e) {

pagesAuthReg->SelectTab(pageReg);

this->Text = "PythonWave: Регистрация";

labelMain->Text = this->Text;

SetCenter(panelMain, labelMain, 1);

}

Void auth::buttonRegister\_Click(Object^ sender, EventArgs^ e) {

if (textBoxLogin1->Text == "") {

textBoxLogin1->BorderColor = Color::Red;

MessageWarning->Show("Введите логин");

return;

}

if (textBoxPassword1->Text == "" || textBoxPassword\_1->Text == "") {

textBoxPassword1->BorderColor = Color::Red;

textBoxPassword\_1->BorderColor = Color::Red;

MessageWarning->Show("Введите пароль");

return;

}

if (textBoxPassword1->Text != textBoxPassword\_1->Text) {

textBoxPassword1->BorderColor = Color::Red;

textBoxPassword\_1->BorderColor = Color::Red;

MessageError->Show("Пароли не совпадают", "Ошибка");

return;

}

if (checkBoxTerms->Checked == false) {

MessageError->Caption = "Предупреждение";

MessageError->Text = "Для регистрации вам нужно прочесть условия использования";

MessageError->Show();

return;

}

if (Directory::Exists(textBoxLogin1->Text)) {

MessageError->Caption = "Предупреждение";

MessageError->Text = "Пользователь под таким логином уже существует";

MessageError->Show();

return;

}

panelMain->Visible = false;

Register();

}

Void auth::Register() {

String^ cliLogin = textBoxLogin1->Text;

String^ cliPassword = textBoxPassword1->Text;

USER = cliLogin;

labelWelcome->Text += cliLogin;

SetCenter(pageWelcome, labelWelcome, 1);

std::string Login = msclr::interop::marshal\_as<std::string>(cliLogin);

std::string Password = msclr::interop::marshal\_as<std::string>(cliPassword);

String^ folderPath = cliLogin;

Directory::CreateDirectory(folderPath);

String^ filePath = Path::Combine(folderPath, "data.bin");

std::hash<std::string> hasher;

size\_t hashPassword = hasher(Password);

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Create, FileAccess::Write);

BinaryWriter^ binaryWriter = gcnew BinaryWriter(fileStream);

binaryWriter->Write(hashPassword);

binaryWriter->Close();

fileStream->Close();

animation\_mode = 1;

sec = 4;

timerTransition->Start();

}

Void auth::linkLabelTerms\_LinkClicked(System::Object^ sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e) {

textBoxTerms->Visible = true;

buttonCloseTerms->Visible = true;

textBoxTerms->BringToFront();

buttonCloseTerms->BringToFront();

}

Void auth::buttonCloseTerms\_Click(System::Object^ sender, System::EventArgs^ e) {

textBoxTerms->Visible = false;

buttonCloseTerms->Visible = false;

}

Void auth::buttonComeIn\_Click(Object^ sender, EventArgs^ e) {

if (textBoxLogin->Text == "") {

textBoxLogin->BorderColor = Color::Red;

MessageWarning->Show("Введите логин");

return;

}

if (textBoxPassword->Text == "") {

textBoxPassword->BorderColor = Color::Red;

MessageWarning->Show("Введите пароль");

return;

}

if (ComeIn()) {

if (SwitchRemember->Checked == true) {

CreateLogFile();

}

if (greeting) {

animation\_mode = 4;

sec = 4;

}

else {

animation\_mode = 5;

sec = 1;

}

labelWelcome->Text += USER;

SetCenter(pageWelcome, labelWelcome, 1);

timerTransition->Start();

}

}

Boolean auth::ComeIn() {

String^ cliLogin = textBoxLogin->Text;

String^ cliPassword = textBoxPassword->Text;

std::string Login = msclr::interop::marshal\_as<std::string>(cliLogin);

std::string Password = msclr::interop::marshal\_as<std::string>(cliPassword);

String^ folderPath = cliLogin;

String^ filePath = Path::Combine(folderPath, "data.bin");

if (!Directory::Exists(folderPath)) {

MessageError->Caption = "Ошибка";

MessageError->Text = "Пользователя с указанным логином не существует";

MessageError->Show();

return false;

}

std::hash<std::string> hasher;

size\_t hashPasswordInput = hasher(Password);

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Open, FileAccess::Read);

BinaryReader^ binaryReader = gcnew BinaryReader(fileStream);

size\_t hashPasswordFile = binaryReader->ReadUInt64();

binaryReader->Close();

fileStream->Close();

if (hashPasswordInput == hashPasswordFile) {

USER = cliLogin;

return true; // Пароль совпадает

}

else {

MessageError->Caption = "Предупреждение";

MessageError->Text = "Введен неверный пароль";

MessageError->Show();

return false; // Пароль не совпадает

}

}

Void auth::timerTransition\_Tick(System::Object^ sender, System::EventArgs^ e) {

ClassFade^ fadeEffect = gcnew ClassFade(this);

switch (animation\_mode)

{

case 1:

--sec;

pagesTransition->SelectTab(pageWelcome);

if (sec <= 0) {

timerTransition->Stop();

sec = 4;

animation\_mode = 2;

timerTransition->Start();

}

break;

case 2:

--sec;

pagesTransition->SelectTab(pageInfo);

if (sec <= 0) {

sec = 2;

animation\_mode = 3;

timerTransition->Start();

}

break;

case 3:

--sec;

if (sec <= 0) {

timerTransition->Stop();

fadeEffect->SetAnimation("hide");

fadeEffect = nullptr;

authToProfile(USER);

}

break;

case 4:

pagesTransition->SelectTab(pageWelcome);

--sec;

if (sec <= 0) {

fadeEffect->SetAnimation("hide");

fadeEffect = nullptr;

timerTransition->Stop();

authToMain(USER);

}

break;

case 5:

--sec;

fadeEffect->SetAnimation("hide");

fadeEffect = nullptr;

if (sec <= 0) {

timerTransition->Stop();

authToMain(USER);

}

break;

}

}

Void auth::textBoxLogin\_Click(System::Object^ sender, System::EventArgs^ e) {

textBoxLogin->BorderColor = Color::White;

}

Void auth::textBoxPassword\_Click(System::Object^ sender, System::EventArgs^ e) {

textBoxPassword->BorderColor = Color::White;

}

void auth::setShadows() {

if (!Directory::Exists("logs")) {

this->borderlessForm->HasFormShadow = true;

}

else if (File::Exists("config.xml")) {

Config^ cfg = gcnew Config();

cfg->LoadConfig();

this->borderlessForm->HasFormShadow = cfg->hasFormShadow;

cfg = nullptr;

}

}

«auth.h»

#pragma once

#include "config.h"

namespace PythonWave {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::IO;

using namespace System::Reflection;

using namespace Guna;

using namespace Guna::UI2;

using namespace Bunifu;

using namespace Bunifu;

public ref class auth : public System::Windows::Forms::Form

{

TabPage^ pageAuth;

TabPage^ pageReg;

LinkLabel^ Ы;

Label^ label1;

Label^ labelRemember;

TabPage^ pageMain;

TabPage^ pageInfo;

Timer^ timerTransition;

TabPage^ pageWelcome;

TabPage^ PageEnd;

Panel^ panelMain;

Label^ labelMain;

IContainer^ components;

WinForms::Guna2CircleButton^ buttonClose;

WinForms::Guna2CircleButton^ buttonMinimize;

WinForms::Guna2ShadowPanel^ panelAuth;

UI::WinForms::BunifuPages^ pagesAuthReg;

WinForms::Guna2Button^ buttonShowAuth;

WinForms::Guna2Button^ buttonShowReg;

WinForms::Guna2Button^ buttonComeIn;

WinForms::Guna2CircleButton^ buttonQuestion;

WinForms::Guna2ToggleSwitch^ SwitchRemember;

WinForms::Guna2TextBox^ textBoxPassword;

WinForms::Guna2TextBox^ textBoxLogin;

WinForms::Guna2Button^ buttonRegister;

WinForms::Guna2TextBox^ textBoxPassword\_1;

WinForms::Guna2TextBox^ textBoxPassword1;

WinForms::Guna2TextBox^ textBoxLogin1;

UI::WinForms::BunifuCheckBox^ checkBoxTerms;

WinForms::Guna2TextBox^ textBoxTerms;

UI::WinForms::BunifuPages^ pagesTransition;

WinForms::Guna2MessageDialog^ MessageError;

WinForms::Guna2CircleButton^ buttonCloseTerms;

WinForms::Guna2PictureBox^ guna2PictureBox1;

WinForms::Guna2Panel^ guna2Panel1;

WinForms::Guna2HtmlLabel^ labelLogo1;

WinForms::Guna2HtmlLabel^ labelWelcome;

WinForms::Guna2PictureBox^ guna2PictureBox3;

WinForms::Guna2PictureBox^ guna2PictureBox4;

WinForms::Guna2MessageDialog^ MessageOption;

WinForms::Guna2DragControl^ dragControl1;

WinForms::Guna2DragControl^ dragControl2;

WinForms::Guna2DragControl^ dragControl3;

WinForms::Guna2HtmlLabel^ guna2HtmlLabel1;

WinForms::Guna2DragControl^ dragControlMain;

WinForms::Guna2DragControl^ dragControlMain2;

WinForms::Guna2MessageDialog^ MessageWarning;

WinForms::Guna2BorderlessForm^ borderlessForm;

public:

auth(void)

{

InitializeComponent();

setShadows();

}

protected:

~auth()

{

if (components)

{

delete components;

}

Application::Exit();

}

#pragma region Windows Form Designer generated code

void InitializeComponent(void){ ... }

#pragma endregion

String^ USER;

private: // Форма

Void LastEnter();

Void auth::loadConfig();

Void buttonMinimize\_Click(System::Object^ sender, System::EventArgs^ e);

Void buttonClose\_Click(System::Object^ sender, System::EventArgs^ e);

Void timerTransition\_Tick(System::Object^ sender, System::EventArgs^ e);

Void buttonShowReg\_Click(System::Object^ sender, System::EventArgs^ e);

Void buttonShowAuth\_Click(System::Object^ sender, System::EventArgs^ e);

private: //Регистрация

Void buttonRegister\_Click(System::Object^ sender, System::EventArgs^ e);

Void Register();

Void buttonCloseTerms\_Click(System::Object^ sender, System::EventArgs^ e);

Void linkLabelTerms\_LinkClicked(System::Object^ sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e);

private: //Вход

Void buttonComeIn\_Click(System::Object^ sender, System::EventArgs^ e);

Boolean ComeIn();

String^ ReadLogFile();

Void CreateLogFile();

Void buttonQuestion\_Click(System::Object^ sender, System::EventArgs^ e);

Void auth\_Load(System::Object^ sender, System::EventArgs^ e);

Void textBoxLogin\_Click(System::Object^ sender, System::EventArgs^ e);

Void textBoxPassword\_Click(System::Object^ sender, System::EventArgs^ e);

void setShadows();

};

}

«profile-header.h»

#pragma once

#include "profile.h"

#include "ClassFade.h"

#include "ClassMail.h"

#include "config.h"

#include <Windows.h>

#include <iostream>

#include <msclr\marshal\_cppstd.h>

#include <string>

#include <map>

#include <ctime>

#using <System.dll>

using namespace System;

using namespace PythonWave;

using namespace System::IO;

using namespace System::Collections::Generic;

using namespace System::Drawing;

using namespace System::Diagnostics;

using namespace System::Net;

using namespace System::Net::Mail;

using namespace System::Net::Security;

using namespace System::Text;

int mode = 1; // Режим перехода

int secondsToClose = 2; // Время до закрытия и для переходов

int secondsLeft = 30; // Повторная отправка письма

int checkBoxMode = 1; // Тип анимации чекбокса

int checkBoxSec = 2; // Время анимации чекбокса

bool email\_confirmed = false; // Состояние подтвердженного email

String^ GetFullDate(String^ day, String^ month, String^ year) {

// Создаем словарь с соответствиями месяцев

Dictionary<String^, String^>^ months = gcnew Dictionary<String^, String^>();

months->Add("Январь", "Января");

months->Add("Февраль", "Февраля");

months->Add("Март", "Марта");

months->Add("Апрель", "Апреля");

months->Add("Май", "Мая");

months->Add("Июнь", "Июня");

months->Add("Июль", "Июля");

months->Add("Август", "Августа");

months->Add("Сентябрь", "Сентября");

months->Add("Октябрь", "Октября");

months->Add("Ноябрь", "Ноября");

months->Add("Декабрь", "Декабря");

// Получаем аббревиатуру месяца из словаря

String^ monthAbbreviation;

if (months->ContainsKey(month)) {

monthAbbreviation = months[month];

}

else {

// Если месяц не найден, оставляем его как есть

monthAbbreviation = month;

}

// Формируем полную дату

String^ fullDate = day + " " + monthAbbreviation + " " + year;

return fullDate;

}

Void SaveDataBinary(String^ FilePath, String^ StringToWrite) {

FileStream^ fileStream = gcnew FileStream(FilePath, FileMode::Create);

BinaryWriter^ binaryWriter = gcnew BinaryWriter(fileStream);

binaryWriter->Write(StringToWrite);

binaryWriter->Close();

fileStream->Close();

}

Void profile::SaveData() {

try {

// Получаем данные

String^ email = textBoxEmail->Text;

String^ name = textBoxUserName->Text;

String^ surname = textBoxUserSurname->Text;

String^ sex = bunifuDropdownSex->Text;

String^ day = bunifuDropdownDay->Text;

String^ month = bunifuDropdownMonth->Text;

String^ year = bunifuDropdownYear->Text;

String^ birth = day + " " + month + " " + year;

String^ lvl = "Новичок";

// Пути сохранения

String^ fileUserEmail = UserLogin + "//userData.bin";

String^ fileUserBirth = UserLogin + "//userBirth.bin";

String^ fileUserSex = UserLogin + "//userSex.bin";

String^ fileUserName = UserLogin + "//userName.bin";

String^ fileUserSurname = UserLogin + "//userSurname.bin";

String^ Rank = UserLogin + "//lvl.bin";

SaveDataBinary(fileUserEmail, email);

SaveDataBinary(fileUserBirth, birth);

SaveDataBinary(fileUserSex, sex);

SaveDataBinary(fileUserName, name);

SaveDataBinary(fileUserSurname, surname);

SaveDataBinary(Rank, lvl);

String^ LogsPath = "script//logs.bin";

Boolean isTaskVisited = false;

SaveDataBinary(LogsPath, Convert::ToString(isTaskVisited));

}

catch (Exception^ e) {

MessageError->Show(e->Message, "Аккаунт будет удален");

}

}

void RestartApplication()

{

String^ executablePath = Application::ExecutablePath;

Process::Start(executablePath);

Application::Exit();

}

int profile::generateSecurityCode() {

srand(time(NULL));

int Code = rand() % 900000 + 100000;

return Code;

}

Void profile::enableMail(bool enable) {

if (enable) {

textBoxCode->Clear();

textBoxEmail->Enabled = true;

textBoxCode->Enabled = false;

buttonSendMail->Enabled = true;

buttonCheckCode->Enabled = false;

pictureBoxCheckMail->Enabled = true;

pictureBoxCheckCode->Enabled = true;

pictureBoxCheckMail->Visible = false;

pictureBoxCheckCode->Visible = false;

linkReMail->Visible = false;

}

else {

textBoxEmail->Enabled = true;

textBoxCode->Enabled = true;

buttonSendMail->Enabled = true;

buttonCheckCode->Enabled = true;

pictureBoxCheckMail->Enabled = false;

pictureBoxCheckCode->Enabled = false;

}

}

Void profile::DeleteDirectory(String^ folderPath) {

try {

DirectoryInfo^ directory = gcnew DirectoryInfo(folderPath);

if (directory->Exists) {

directory->Delete(true);

}

else {

MessageError->Show("Аккаунта не существует");

}

}

catch (Exception^ e) {

MessageError->Show(e->Message);

}

}

bool profile::IsValidEmail(String^ email) {

return email->Contains("@") && email->Contains(".");

}

Void profile::profile\_Load(System::Object^ sender, System::EventArgs^ e) {

SetCenter(pageSettings, labelSettings, 1);

SetCenter(pageSettings, labelClose, 1);

SetCenter(pageSettings, labelEnd, 3);

SetCenter(pageSettings, labelEnd, 3);

SetCenter(pageSettings, labelClose, 1);

SetCenter(panel1, labelMain, 1);

SetCenter(pageEnd, labelEnd, 3);

SetCenter(pageEnd, labelClose, 1);

ClassFade^ Fade = gcnew ClassFade(this);

Fade->SetAnimation("in");

Fade = nullptr;

}

Void profile::ButtonMinimize\_Click(System::Object^ sender, System::EventArgs^ e) {

ClassFade^ Fade = gcnew ClassFade(this);

Fade->SetAnimation("minimize");

Fade = nullptr;

}

Void profile::ButtonExit\_Click(System::Object^ sender, System::EventArgs^ e) {

Windows::Forms::DialogResult result = MessageDialogExit->Show();

if (result == Windows::Forms::DialogResult::Yes) {

DeleteDirectory(UserLogin);

ClassFade^ Fade = gcnew ClassFade(this);

Fade->SetAnimation("close");

Fade = nullptr;

}

}

Void profile::buttonQuestion\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageDialogQuestion->Show();

}

Void profile::timerTransition\_Tick(System::Object^ sender, System::EventArgs^ e) {

switch (mode)

{

case 1:

--secondsToClose;

guna2CustomCheckBox1->Checked = true;

if (secondsToClose == 0) {

timerTransition->Stop();

mode = 2;

secondsToClose = 5;

Pages->SelectTab(pageSettings);

}

break;

case 2:

if (secondsToClose == 0) {

ClassFade^ Fade = gcnew ClassFade(this);

Fade->SetAnimation("Hide");

Fade = nullptr;

RestartApplication();

}

else {

SetCenter(pageEnd, labelClose, 1);

--secondsToClose;

labelClose->Text = "Приложение перезапустится через " + secondsToClose + " секунд";

}

break;

} {

}

}

Void profile::timer\_Tick(System::Object^ sender, System::EventArgs^ e) {

secondsLeft--;

if (secondsLeft <= 0)

{

secondsLeft = 30;

timerReMail->Stop();

labelTimer->Visible = false;

linkReMail->Visible = true;

linkReMail->Enabled = true;

}

else

{

labelTimer->Text = "Отправить повторно через " + secondsLeft.ToString() + " секунд";

}

}

Void profile::textBoxMail\_Click(System::Object^ sender, System::EventArgs^ e) {

textBoxEmail->BorderColor = Color::Blue;

}

Void profile::textBoxUserName\_Click(System::Object^ sender, EventArgs^ e) {

textBoxUserName->BorderColor = Color::Gray;

}

Void profile::bunifuDropdownSex\_Click(System::Object^ sender, EventArgs^ e) {

bunifuDropdownSex->BorderColor = Color::Gray;

}

Void profile::bunifuDropdownDay\_Click(System::Object^ sender, EventArgs^ e) {

bunifuDropdownDay->BorderColor = Color::Gray;

}

Void profile::bunifuDropdownMonth\_Click(System::Object^ sender, EventArgs^ e) {

bunifuDropdownMonth->BorderColor = Color::Gray;

}

Void profile::bunifuDropdownYear\_Click(System::Object^ sender, EventArgs^ e) {

bunifuDropdownYear->BorderColor = Color::Gray;

}

Void profile::linkReMail\_Click(System::Object^ sender, System::EventArgs^ e) {

enableMail(true);

email\_confirmed = false;

}

Void profile::buttonResume\_Click(System::Object^ sender, System::EventArgs^ e) {

bool confirmed = true;

if (textBoxUserName->Text == "") {

textBoxUserName->BorderColor = Color::Red;

confirmed = false;

}

if (bunifuDropdownSex->Text == "Пол") {

bunifuDropdownSex->BorderColor = Color::Red;

confirmed = false;

}

if (bunifuDropdownDay->Text == "День") {

bunifuDropdownDay->BorderColor = Color::Red;

confirmed = false;

}

if (bunifuDropdownMonth->Text == "Месяц") {

bunifuDropdownMonth->BorderColor = Color::Red;

confirmed = false;

}

if (bunifuDropdownYear->Text == "Год") {

bunifuDropdownMonth->BorderColor = Color::Red;

confirmed = false;

}

if (!confirmed) {

MessageError->Show("Заполните все необходимые данные");

return;

}

else if (!email\_confirmed) {

MessageWarning->Show("Подтвердите email");

textBoxEmail->BorderColor = Color::Red;

return;

}

SaveData();

succesful = true;

Pages->SelectTab(pageSuccessful);

timerTransition->Start();

}

Void profile::buttonSendMail\_Click(System::Object^ sender, System::EventArgs^ e) {

ClassMail^ EMAIL = gcnew ClassMail(this);

String^ userMail = Convert::ToString(textBoxEmail->Text);

String^ mail = "Здравствуйте, ваш код регистрации: " + Convert::ToString(SecurityCode);

if (IsValidEmail(userMail)) {

if (EMAIL->SendEmail(userMail, "PythonWave: Код безопасности", mail)) {

textBoxEmail->Enabled = false;

buttonSendMail->Enabled = false;

textBoxCode->Enabled = true;

buttonCheckCode->Enabled = true;

pictureBoxCheckMail->Visible = true;

labelTimer->Visible = true;

timerReMail->Start();

}

}

else {

MessageError->Caption = "Письмо не было отправлено";

MessageError->Text = "Введите корректный адрес эл. почты";

MessageError->Show();

textBoxEmail->BorderColor = Color::Red;

textBoxEmail->Clear();

}

}

void profile::SaveConfig() {

Config^ config = gcnew Config();

config->borderForm = TrackBorderForm->Value;

config->borderBtn = TrackBorderBtn->Value;

config->volume = TrackVolume->Value;

config->dragTransparent = toggleTransparent->Checked;

config->greeting = toggleGreeting->Checked;

config->hasFormShadow = toggleShadows->Checked;

config->SaveConfig();

}

Void profile::buttonResume1\_Click(System::Object^ sender, System::EventArgs^ e) {

SaveConfig();

Pages->SelectTab(pageEnd);

mode = 2;

timerTransition->Start();

}

Void profile::buttonValidateCode\_Click(System::Object^ sender, System::EventArgs^ e) {

try {

if (SecurityCode == Convert::ToInt32(textBoxCode->Text)) {

email\_confirmed = true;

textBoxCode->Enabled = false;

buttonCheckCode->Enabled = false;

pictureBoxCheckCode->Visible = true;

timerReMail->Stop();

labelTimer->Visible = false;

linkReMail->Visible = false;

}

else {

MessageError->Text = "Неверный код";

MessageError->Show();

}

}

catch (Exception^ e) {

MessageError->Show(e->Message);

}

}

Void profile::buttonUpload\_Click(System::Object^ sender, System::EventArgs^ e) {

try {

openFileDialog1->Filter = "Image Files (\*.jpg;\*.png)|\*.jpg;\*.png";

openFileDialog1->FilterIndex = 1;

openFileDialog1->RestoreDirectory = true;

if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {

String^ filePath = openFileDialog1->FileName;

Drawing::Image^ image = System::Drawing::Image::FromFile(filePath);

pictureBoxUploadImage->SizeMode = PictureBoxSizeMode::StretchImage;

pictureBoxUploadImage->Image = image;

Drawing::Imaging::ImageFormat^ imageFormat = image->RawFormat;

String^ savePath = UserLogin + "\\avatar";

if (imageFormat->Equals(System::Drawing::Imaging::ImageFormat::Jpeg)) {

savePath += ".jpg";

image->Save(savePath, System::Drawing::Imaging::ImageFormat::Jpeg);

}

else if (imageFormat->Equals(System::Drawing::Imaging::ImageFormat::Png)) {

savePath += ".png";

image->Save(savePath, System::Drawing::Imaging::ImageFormat::Png);

}

pictureBoxUploadImage->Image->Save(savePath, imageFormat);

}

}

catch (Exception^ ex) {

MessageDialogUpload->Show(ex->Message);

}

}

Void profile::buttonCloseEnd\_Click(System::Object^ sender, System::EventArgs^ e) {

ClassFade^ Fade = gcnew ClassFade(this);

Fade->SetAnimation("close");

Fade = nullptr;

}

Void profile::guna2TrackBar1\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e) {

borderlessForm->BorderRadius = TrackBorderForm->Value;

labelBorderForm->Text = Convert::ToString(TrackBorderForm->Value);

}

Void profile::guna2ToggleSwitch1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

dragSettings->TransparentWhileDrag = toggleTransparent->Checked;

}

Void profile::toggleShadows\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

borderlessForm->HasFormShadow = toggleShadows->Checked;

if (borderlessForm->HasFormShadow == false) {

MessageInfo->Show("Изменения вступят в силу после перезапуска");

}

}

Void profile::guna2CircleButton1\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageInfo->Show("Эта функция на стадии ранней разработки");

}

Void profile::guna2TrackBar2\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e) {

labelVolume->Text = Convert::ToString(TrackVolume->Value);

}

Void profile::guna2TrackBar3\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e) {

buttonResume1->BorderRadius = TrackBorderBtn->Value;

labelBorderBtn->Text = Convert::ToString(TrackBorderBtn->Value);

}

«profile.h»

#pragma once

namespace PythonWave {

using namespace System;

using namespace System::Windows;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

public ref class profile : public System::Windows::Forms::Form

{

private:

Guna::UI2::WinForms::Guna2BorderlessForm^ borderlessForm;

Bunifu::UI::WinForms::BunifuPages^ Pages;

System::Windows::Forms::TabPage^ pageProfile;

Guna::UI2::WinForms::Guna2Button^ buttonResume;

Guna::UI2::WinForms::Guna2CirclePictureBox^ buttonClose;

Guna::UI2::WinForms::Guna2CirclePictureBox^ buttonMin;

System::Windows::Forms::PictureBox^ pictureBoxUploadImage;

Guna::UI2::WinForms::Guna2Button^ buttonUpload;

Guna::UI2::WinForms::Guna2TextBox^ textBoxEmail;

System::Windows::Forms::LinkLabel^ linkReMail;

Guna::UI2::WinForms::Guna2TextBox^ textBoxCode;

Guna::UI2::WinForms::Guna2Button^ buttonSendMail;

Guna::UI2::WinForms::Guna2CircleButton^ buttonQuestion;

Guna::UI2::WinForms::Guna2Button^ buttonCheckCode;

System::Windows::Forms::Label^ label3;

System::Windows::Forms::Label^ label1;

Guna::UI2::WinForms::Guna2TextBox^ textBoxUserSurname;

System::Windows::Forms::Label^ label2;

Guna::UI2::WinForms::Guna2TextBox^ textBoxUserName;

System::Windows::Forms::TabPage^ pageSuccessful;

Guna::UI2::WinForms::Guna2CustomCheckBox^ guna2CustomCheckBox1;

System::Windows::Forms::TabPage^ pageSettings;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageDialogExit;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageDialogQuestion;

System::Windows::Forms::OpenFileDialog^ openFileDialog1;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageDialogUpload;

System::Windows::Forms::Timer^ timerTransition;

System::Windows::Forms::Panel^ panel1;

System::Windows::Forms::Label^ labelMain;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageError;

System::Windows::Forms::TabPage^ pageEnd;

Guna::UI2::WinForms::Guna2Button^ buttonResume1;

Bunifu::UI::WinForms::BunifuLabel^ bunifuLabel1;

Bunifu::UI::WinForms::BunifuLabel^ labelSettings;

Bunifu::UI::WinForms::BunifuLabel^ labelEnd;

System::Windows::Forms::Panel^ panel2;

Bunifu::UI::WinForms::BunifuLabel^ labelClose;

System::Windows::Forms::Timer^ timerCheckBox;

Guna::UI2::WinForms::Guna2CirclePictureBox^ pictureBoxCheckCode;

Guna::UI2::WinForms::Guna2CirclePictureBox^ pictureBoxCheckMail;

Guna::UI2::WinForms::Guna2DragControl^ dragControlPanel;

Guna::UI2::WinForms::Guna2DragControl^ dragControlProfile;

System::Windows::Forms::Label^ labelTimer;

System::Windows::Forms::Timer^ timerReMail;

Bunifu::UI::WinForms::BunifuDropdown^ bunifuDropdownSex;

Bunifu::UI::WinForms::BunifuDropdown^ bunifuDropdownDay;

Bunifu::UI::WinForms::BunifuDropdown^ bunifuDropdownMonth;

Bunifu::UI::WinForms::BunifuDropdown^ bunifuDropdownYear;

System::Windows::Forms::Label^ label4;

Guna::UI2::WinForms::Guna2DragControl^ dragSettings;

Guna::UI2::WinForms::Guna2DragControl^ dragEnd;

Guna::UI2::WinForms::Guna2DragControl^ dragSuccessful;

Guna::UI2::WinForms::Guna2DragControl^ dragControlPanel2;

Guna::UI2::WinForms::Guna2TrackBar^ TrackBorderForm;

Guna::UI2::WinForms::Guna2ToggleSwitch^ toggleTransparent;

Guna::UI2::WinForms::Guna2Separator^ guna2Separator1;

Bunifu::UI::WinForms::BunifuLabel^ bunifuLabel2;

System::Windows::Forms::Label^ label6;

System::Windows::Forms::Label^ label5;

System::Windows::Forms::Label^ label7;

Guna::UI2::WinForms::Guna2ToggleSwitch^ toggleShadows;

Guna::UI2::WinForms::Guna2CircleButton^ guna2CircleButton1;

System::Windows::Forms::Label^ label8;

Guna::UI2::WinForms::Guna2ToggleSwitch^ toggleGreeting;

System::Windows::Forms::Label^ labelBorderForm;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageInfo;

System::Windows::Forms::Label^ labelVolume;

System::Windows::Forms::Label^ label10;

Guna::UI2::WinForms::Guna2TrackBar^ TrackVolume;

System::Windows::Forms::Label^ labelBorderBtn;

System::Windows::Forms::Label^ label12;

Guna::UI2::WinForms::Guna2TrackBar^ TrackBorderBtn;

Bunifu::UI::WinForms::BunifuLabel^ bunifuLabel3;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageWarning;

Guna::UI2::WinForms::Guna2BorderlessForm^ guna2BorderlessForm1;

System::ComponentModel::IContainer^ components;

public:

String^ UserLogin;

profile(String^ login)

{

InitializeComponent();

UserLogin = login;

srand(static\_cast<unsigned int>(time(NULL)));

}

profile(void)

{

InitializeComponent();

MessageError->Show("Пользователь не был передан в форму");

Application::Exit();

}

protected:

~profile()

{

if (!succesful) {

MessageError->Show("Ваш аккаунт будет удален", "Ошибка регистрации");

DeleteDirectory(UserLogin);

}

if (components)

{

delete components;

}

Application::Exit();

}

#pragma region Windows Form Designer generated code

void InitializeComponent(void){ ... }

#pragma endregion

private:

Boolean succesful = false;

Void profile\_Load(Object^ sender, EventArgs^ e);

Void SaveData();

Void SaveConfig();

int generateSecurityCode();

bool IsValidEmail(String^ email);

int SecurityCode = generateSecurityCode();

Void enableMail(bool enable);

Void DeleteDirectory(String^ folderPath);

private:

Void buttonSendMail\_Click(Object^ sender, EventArgs^ e);

Void buttonValidateCode\_Click(Object^ sender, EventArgs^ e);

Void ButtonMinimize\_Click(Object^ sender, EventArgs^ e);

Void ButtonExit\_Click(Object^ sender, EventArgs^ e);

Void buttonQuestion\_Click(Object^ sender, EventArgs^ e);

Void buttonUpload\_Click(Object^ sender, EventArgs^ e);

Void buttonResume\_Click(Object^ sender, EventArgs^ e);

Void buttonResume1\_Click(Object^ sender, EventArgs^ e);

Void textBoxMail\_Click(Object^ sender, EventArgs^);

private:

Void timer\_Tick(Object^ sender, EventArgs^ e);

Void timerTransition\_Tick(Object^ sender, EventArgs^ e);

Void linkReMail\_Click(System::Object^ sender, EventArgs^ e);

Void textBoxUserName\_Click(System::Object^ sender, EventArgs^ e);

Void bunifuDropdownSex\_Click(System::Object^ sender, EventArgs^ e);

Void bunifuDropdownDay\_Click(System::Object^ sender, EventArgs^ e);

Void bunifuDropdownMonth\_Click(System::Object^ sender, EventArgs^ e);

Void bunifuDropdownYear\_Click(System::Object^ sender, EventArgs^ e);

Void buttonCloseEnd\_Click(System::Object^ sender, System::EventArgs^ e);

private:

Void guna2TrackBar1\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e);

Void guna2ToggleSwitch1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e);

Void toggleShadows\_CheckedChanged(System::Object^ sender, System::EventArgs^ e);

Void guna2CircleButton1\_Click(System::Object^ sender, System::EventArgs^ e);

Void guna2TrackBar2\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e);

Void guna2TrackBar3\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e);

};

}

«main-header.h»

#pragma once

#include "MainMenu.h"

#include "MainBook.h"

#include "MainFunc.h"

#include "MainProfile.h"

#include "MainTasks.h"

#include "MainSettings.h"

#include "MainHome.h"

#include <random>

bool test = true;

bool messageNeed = false;

void SetCenter(Control^ background, Control^ control, int mode);

void mainForm::Form\_MouseDown(Object^ sender, MouseEventArgs^ e) {

if (alwaysHideMenu) {

menu = false;

timerMenu->Start();

}

}

Void mainForm::main\_Load(System::Object^ sender, System::EventArgs^ e) {

isDataLoaded = false;

menu = false;

DataLoad();

cfgLoad();

logsLoad();

LoadBook();

LoadHomePage();

ClassFade^ Fade = gcnew ClassFade(this);

Fade->SetAnimation("in");

AppTimer->Start();

SetCenter(panelProfileData, lblLogin, 1);

SetCenter(anim1, lblAnim1, 1);

SetCenter(pageProfile, pictureProfile, 1);

SetCenter(pageProfile, panelProfileData, 1);

isDataLoaded = true;

}

Void mainForm::btnExit\_Click(System::Object^ sender, System::EventArgs^ e) {

ClassFade^ Fade = gcnew ClassFade(this);

Fade->SetAnimation("close");

}

Void mainForm::btnMinimize\_Click(System::Object^ sender, System::EventArgs^ e) {

ClassFade^ Fade = gcnew ClassFade(this);

Fade->SetAnimation("minimize");

}

Void mainForm::btnMaximize\_Click(System::Object^ sender, System::EventArgs^ e) {

if (this->WindowState == FormWindowState::Normal) {

this->WindowState = FormWindowState::Maximized;

}

else {

this->WindowState = FormWindowState::Normal;

this->Size = Drawing::Size(1600, 900);

}

}

Void mainForm::ratingUser\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageInfo->Caption = "Информация";

MessageInfo->Show("Оценка ваших успехов в учебе :)", "Звездочки");

}

Void mainForm::btnQuestionBook\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageInfo->Show("Примените желаемые пометки, выделив текст", "Панель инструментов");

}

String^ GetRandomTask(array<String^>^ tasksArray) {

Random^ rand = gcnew Random();

int index = rand->Next(tasksArray->Length);

return tasksArray[index];

}

Void mainForm::AppTimer\_Tick(System::Object^ sender, System::EventArgs^ e) {

array<String^>^ EzTasks = { "add", "multiply", "divide", "subtract", "even\_or\_odd" };

array<String^>^ MidTasks = { "better\_than\_average", "positive\_sum", "reverse\_seq" };

array<String^>^ HardTasks = { "get\_count", "high\_and\_low", "square\_digits" };

array<String^>^ VeryHardTasks = { "get\_char", "symmetric\_point", "get\_middle" };

if (hacks) {

secondsSpent += 5000;

ClassProgress data(User);

ClassTasks tasks(User);

if (!test) {

if (messageNeed) {

MessageInfo->Show("Все задачи пройдены", "Тестирование");

messageNeed = false;

}

}

else if (secondsSpent >= 70000) {

messageNeed = true;

test = false;

data.SolveTaskSplus();

tasks.SetTaskCompletionDate(GetRandomTask(VeryHardTasks));

}

if (secondsSpent >= 55000) {

data.SolveTaskS();

tasks.SetTaskCompletionDate(GetRandomTask(HardTasks));

}

else if (secondsSpent >= 40000) {

data.SolveTaskB();

tasks.SetTaskCompletionDate(GetRandomTask(MidTasks));

}

else if (secondsSpent >= 25000) {

data.SolveTaskA();

tasks.SetTaskCompletionDate(GetRandomTask(EzTasks));

}

}

else

++secondsSpent;

WriteTimeToFile();

ReadTimeFromFile();

DataLoad();

LoadHomePage();

}

Void mainForm::guna2Button1\_Click(System::Object^ sender, System::EventArgs^ e) {

hacks == false ? hacks = true : hacks = false;

if (hacks) {

MessageInfo->Show("Тестирование приложения включено", "Информация");

ClassProgress data(User);

ClassTasks tasks(User);

data.SolveTaskA();

tasks.SetTaskCompletionDate("add");

}

else {

MessageInfo->Show("Тестирование приложения выключено", "Информация");

}

DataLoad();

LoadHomePage();

}

Void mainForm::mainForm\_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {

WriteTimeToFile();

}

Void mainForm::mainForm\_Resize(System::Object^ sender, System::EventArgs^ e) {

this->Size = Drawing::Size(1600, 900);

}

«MainMenu.h»

#pragma once

#include "mainForm.h"

using namespace PythonWave;

using namespace System;

Void mainForm::SetProfileDefaults() {

textBoxUserName->Text = UserName;

textBoxUserSurname->Text = UserSurname;

textBoxEmail->Text = UserEmail;

textBoxCode->Enabled = false;

textBoxCodeNew->Enabled = false;

buttonCheckCode->Enabled = false;

}

Boolean mainForm::CheckSave() {

if (isProfileSaved) {

return true;

}

else {

MessageWarning->Buttons = UI2::WinForms::MessageDialogButtons::YesNo;

Forms::DialogResult result = MessageWarning->Show("У вас остались несохраненные изменения", "Действительно хотите перейти?");

if (result == Windows::Forms::DialogResult::Yes) {

isProfileSaved = true;

SetProfileDefaults();

return true;

}

else {

return false;

}

}

MessageWarning->Buttons = UI2::WinForms::MessageDialogButtons::OK;

}

Void mainForm::timerMenu\_Tick(System::Object^ sender, System::EventArgs^ e) {

int newPanelWidth;

int newPagesWidth;

int newPagesLocationX;

switch (menu)

{

case true:

if (panelMenu->Size.Width < 250) {

int newPanelWidth = panelMenu->Size.Width + 10;

int newPagesWidth = Pages->Size.Width - 10;

int newPagesLocationX = Pages->Location.X + 10;

panelMenu->Width = newPanelWidth;

Pages->Width = newPagesWidth;

Pages->Location = Drawing::Point(newPagesLocationX, Pages->Location.Y);

}

else {

newPanelWidth = NULL;

newPagesWidth = NULL;

newPagesLocationX = NULL;

timerMenu->Stop();

}

break;

case false:

if (panelMenu->Size.Width > 80) {

int newPanelWidth = panelMenu->Size.Width - 10;

int newPagesWidth = Pages->Size.Width + 10;

int newPagesLocationX = Pages->Location.X - 10;

panelMenu->Width = newPanelWidth;

Pages->Width = newPagesWidth;

Pages->Location = Drawing::Point(newPagesLocationX, Pages->Location.Y);

}

else {

newPanelWidth = NULL;

newPagesWidth = NULL;

newPagesLocationX = NULL;

timerMenu->Stop();

}

break;

default:

break;

}

}

Void mainForm::funcSelectTab(TabPage^ tabPage) {

LoadHomePage();

if (CheckSave() && !courseAnimationState)

Pages->SelectTab(tabPage);

else if (courseAnimationState)

btnCourses->Checked = true;

}

Void mainForm::btnMenu\_Click(System::Object^ sender, System::EventArgs^ e) {

if (courseAnimationState)

return;

menu == false ? menu = true : menu = false;

timerMenu->Start();

}

Void mainForm::btnHome\_Click(System::Object^ sender, System::EventArgs^ e) {

funcSelectTab(pageHome);

}

Void mainForm::btnBook\_Click(System::Object^ sender, System::EventArgs^ e) {

funcSelectTab(pageBook);

}

Void mainForm::btnProfile\_Click(System::Object^ sender, System::EventArgs^ e) {

logsLoad();

funcSelectTab(pageProfile);

btnProfile->Checked = true;

}

Void mainForm::btnStats\_Click(System::Object^ sender, System::EventArgs^ e) {

funcSelectTab(pageStats);

}

Void mainForm::btnSettings\_Click(System::Object^ sender, System::EventArgs^ e) {

funcSelectTab(pageSettings);

}

Void mainForm::btnLogOut\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ executablePath = Application::ExecutablePath;

Process::Start(executablePath);

if (Directory::Exists("logs")) {

try {

Directory::Delete("logs", true);

}

catch (Exception^ e) {

MessageError->Show(e->Message);

}

}

Application::Exit();

}

«MainBook.h»

#pragma once

#include "mainForm.h"

using namespace System;

Boolean isBookExists = true;

void mainForm::SaveBook(String^ Point, RichTextBox^ richTextBox) {

String^ filePath = User + "//book//" + Point + ".rtf";

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Create, FileAccess::Write);

StreamWriter^ sw = gcnew StreamWriter(fileStream);

sw->Write(richTextBox->Rtf);

sw->Close();

}

void mainForm::LoadPage(String^ Point, RichTextBox^ richTextBox) {

String^ filePath = User + "//book//" + Point + ".rtf";

if (!File::Exists(filePath)) {

filePath = "book//" + Point + ".rtf";

}

if (!File::Exists(filePath)) {

MessageBox::Show("Файл:" + filePath + "Не найден", "Ошибка");

isBookExists = false;

return;

}

try {

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Open, FileAccess::Read, FileShare::ReadWrite);

StreamReader^ sr = gcnew StreamReader(fileStream);

String^ rtfContent = sr->ReadToEnd();

sr->Close();

fileStream->Close();

richTextBox->Rtf = rtfContent;

}

catch (IOException^ e) {

MessageError->Show(e->Message, "Ошибка загрузки книги");

}

}

void mainForm::LoadBook() {

LoadPage("start", richStart);

array<RichTextBox^>^ richTextBoxes = { richBook1, richBook2, richBook3, richBook4, richBook5, richBook6, richBook7, richBook8, richBook9, richBook10, richBook11, richBook12, richBook13 };

int countOfPages = Book->TabCount - 2;

for (int i = 0; i < countOfPages; ++i) {

String^ str = "r" + Convert::ToString(i + 1);

LoadPage(str, richTextBoxes[i]);

}

String^ filePath = User + "//book//lastpage.txt";

if (File::Exists(filePath)) {

String^ pageName = File::ReadAllText(filePath);

int tabPageIndex = -1;

// Определяем соответствующий индекс вкладки для имени страницы

for (int i = 0; i < Book->TabCount; ++i) {

if (Book->TabPages[i]->Text == pageName) {

tabPageIndex = i;

break;

}

}

if (tabPageIndex != -1) {

Book->SelectTab(tabPageIndex);

dropdownPages->SelectedIndex = tabPageIndex;

}

}

if (!isBookExists)

MessageWarning->Show("Ошибка загрузки книги", "Файлы повреждены");

}

void mainForm::UpdateLastBookPage() {

CurrentBookPage = Book->SelectedTab;

CurrentBookIndex = Book->SelectedIndex;

CurrentBookFile = CurrentBookPage->Text;

for each (Control ^ control in CurrentBookPage->Controls) {

RichTextBox^ richTextBox = dynamic\_cast<RichTextBox^>(control);

if (richTextBox != nullptr) {

CurrentRichBox = richTextBox;

break;

}

}

}

bool mainForm::isCustomBookExists() {

return Directory::Exists(User + "//myBook");

}

bool mainForm::isLastPageExists() {

return File::Exists(User + "//myBook//lastpage.txt");

}

void mainForm::CreateLastPage() {

if (!isLastPageExists()) {

try {

String^ filePath = User + "//book//lastpage.txt";

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Create, FileAccess::Write);

StreamWriter^ writer = gcnew StreamWriter(fileStream);

writer->Write(CurrentBookPage->Text);

writer->Close();

fileStream->Close();

}

catch (Exception^ e) {

MessageError->Show(e->Message);

}

}

}

void mainForm::CreateCustomBook() {

if (!isCustomBookExists())

Directory::CreateDirectory(User + "//book");

}

void mainForm::ChangeSelectionBackColor(Color color, RichTextBox^ richTB) {

UpdateLastBookPage();

CreateCustomBook();

if (richTB->SelectionLength <= 0) {

MessageInfo->Show("Выделите текст", "Выделение текста");

return;

}

richTB->SelectionBackColor = color;

}

void mainForm::ChangeSelectionFontStyle(FontStyle style, RichTextBox^ richTB)

{

UpdateLastBookPage();

CreateCustomBook();

if (richTB->SelectionLength <= 0) {

MessageInfo->Show("Выделите текст", "Изменение шрифта");

return;

}

Drawing::Font^ currentFont = richTB->SelectionFont;

Drawing::Font^ newFont = gcnew Drawing::Font(currentFont, currentFont->Style ^ style);

richTB->SelectionFont = newFont;

}

void mainForm::SetSelectionFontByDefault(RichTextBox^ richTB)

{

UpdateLastBookPage();

CreateCustomBook();

if (richTB->SelectionLength <= 0) {

if (MessageInfo != nullptr)

MessageInfo->Show("Выделите текст", "Очистка пометок");

return;

}

try {

int start = richTB->SelectionStart;

int end = start + richTB->SelectionLength;

for (int i = start; i < end; i++) {

Drawing::Font^ currentFont = richTB->SelectionFont;

if (currentFont == nullptr) {

currentFont = richTB->Font;

}

Color currentBackColor = richTB->SelectionBackColor;

}

Drawing::Font^ defaultFont = gcnew Drawing::Font("Century Gothic", richTB->SelectionFont->Size);

richTB->SelectionFont = defaultFont;

richTB->SelectionBackColor = Color::FromArgb(64, 66, 88);

}

catch (Exception^ e) {

if (MessageError != nullptr)

MessageError->Show(e->Message, "Неизвестная ошибка");

}

}

Void mainForm::btnHighlight\_Click(System::Object^ sender, System::EventArgs^ e) {

UpdateLastBookPage();

ChangeSelectionBackColor(Color::Gray, CurrentRichBox);

SaveBook(CurrentBookFile, CurrentRichBox);

}

Void mainForm::btnUnderline\_Click(System::Object^ sender, System::EventArgs^ e) {

UpdateLastBookPage();

ChangeSelectionFontStyle(FontStyle::Underline, CurrentRichBox);

SaveBook(CurrentBookFile, CurrentRichBox);

}

Void mainForm::btnThick\_Click(System::Object^ sender, System::EventArgs^ e) {

UpdateLastBookPage();

ChangeSelectionFontStyle(FontStyle::Bold, CurrentRichBox);

SaveBook(CurrentBookFile, CurrentRichBox);

}

Void mainForm::btnStrikeOut\_Click(System::Object^ sender, System::EventArgs^ e) {

UpdateLastBookPage();

ChangeSelectionFontStyle(FontStyle::Strikeout, CurrentRichBox);

SaveBook(CurrentBookFile, CurrentRichBox);

}

Void mainForm::btnItalic\_Click(System::Object^ sender, System::EventArgs^ e) {

UpdateLastBookPage();

ChangeSelectionFontStyle(FontStyle::Italic, CurrentRichBox);

SaveBook(CurrentBookFile, CurrentRichBox);

}

Void mainForm::btnClearFilters\_Click(System::Object^ sender, System::EventArgs^ e) {

UpdateLastBookPage();

SetSelectionFontByDefault(CurrentRichBox);

SaveBook(CurrentBookFile, CurrentRichBox);

}

Void mainForm::dropdownPages\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

int selectedIndex = dropdownPages->SelectedIndex;

Book->SetPage(selectedIndex);

UpdateLastBookPage();

CreateCustomBook();

CreateLastPage();

}

Void mainForm::btnPreviousBook\_Click(System::Object^ sender, System::EventArgs^ e) {

int currentIndex = dropdownPages->SelectedIndex;

if (currentIndex > 0) {

dropdownPages->SelectedIndex = currentIndex - 1;

}

}

Void mainForm::btnBookNext\_Click(System::Object^ sender, System::EventArgs^ e) {

int currentIndex = dropdownPages->SelectedIndex;

if (currentIndex < dropdownPages->Items->Count - 1) {

dropdownPages->SelectedIndex = currentIndex + 1;

}

}

«MainFunc.h»

#pragma once

#include "mainForm.h"

#include "ClassProgress.h"

void mainForm::logsLoad() {

if (File::Exists("script//logs.bin"))

isCoursesVisited = Convert::ToBoolean(readBinaryFile("script//logs.bin"));

else

isCoursesVisited = false;

}

Void mainForm::enableMail(bool enable) {

if (enable) {

textBoxCode->Clear();

textBoxEmail->Enabled = true;

textBoxCode->Enabled = false;

textBoxCodeNew->Enabled = false;

buttonSendMail->Enabled = true;

buttonCheckCode->Enabled = false;

pictureBoxCheckMail->Enabled = true;

pictureBoxCheckCode->Enabled = true;

pictureBoxCheckMail->Visible = false;

pictureBoxCheckCode->Visible = false;

linkReMail->Visible = false;

}

else {

textBoxEmail->Enabled = true;

textBoxCode->Enabled = true;

buttonSendMail->Enabled = true;

buttonCheckCode->Enabled = true;

pictureBoxCheckMail->Enabled = false;

pictureBoxCheckCode->Enabled = false;

}

}

String^ mainForm::GetFolderCreationDate(String^ folderPath) {

pin\_ptr<const wchar\_t> wch = PtrToStringChars(folderPath);

const std::wstring folderPathStr(wch);

WIN32\_FILE\_ATTRIBUTE\_DATA fileInfo;

if (GetFileAttributesEx(folderPathStr.c\_str(), GetFileExInfoStandard, &fileInfo)) {

FILETIME creationTime = fileInfo.ftCreationTime;

SYSTEMTIME systemTime;

FileTimeToSystemTime(&creationTime, &systemTime);

array<String^>^ months = gcnew array<String^> {

"Января", "Февраля", "Марта", "Апреля", "Мая", "Июня",

"Июля", "Августа", "Сентября", "Октября", "Ноября", "Декабря"

};

String^ creationDateString = String::Format("{0} {1} {2}",

systemTime.wDay, months[systemTime.wMonth - 1], systemTime.wYear);

return creationDateString;

}

else {

return "Ошибка получения даты";

}

}

String^ mainForm::readBinaryFile(String^ filePath) {

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Open, FileAccess::Read);

BinaryReader^ binaryReader = gcnew BinaryReader(fileStream);

String^ result = binaryReader->ReadString();

fileStream->Close();

binaryReader->Close();

return result;

}

size\_t mainForm::readPassword(String^ filePath) {

try {

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Open, FileAccess::Read);

BinaryReader^ binaryReader = gcnew BinaryReader(fileStream);

size\_t result = binaryReader->ReadUInt64();

binaryReader->Close();

fileStream->Close();

return result;

}

catch (Exception^ e) {

MessageError->Show("0: " + e->Message);

return 0;

}

}

void mainForm::writeBinaryFile(String^ filePath, String^ content) {

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Create, FileAccess::Write);

BinaryWriter^ binaryWriter = gcnew BinaryWriter(fileStream);

binaryWriter->Write(content);

binaryWriter->Close();

fileStream->Close();

}

bool mainForm::IsValidEmail(String^ email) {

return email->Contains("@") && email->Contains(".");

}

int mainForm::generateSecurityCode() {

int Code = rand() % 900000 + 100000;

return Code;

}

void mainForm::DeleteDirectory(String^ folderPath) {

try {

DirectoryInfo^ directory = gcnew DirectoryInfo(folderPath);

if (directory->Exists) {

directory->Delete(true);

}

else {

MessageError->Show("Аккаунт не найден");

}

}

catch (Exception^ e) {

MessageError->Show(e->Message);

}

}

void mainForm::DataLoad() {

ClassProgress data(User);

String^ fileUserEmail = User + "//userData.bin";

String^ fileUserBirth = User + "//userBirth.bin";

String^ fileUserSex = User + "//userSex.bin";

String^ fileUserName = User + "//userName.bin";

String^ fileUserSurname = User + "//userSurname.bin";

try {

UserEmail = readBinaryFile(fileUserEmail);

UserBirth = readBinaryFile(fileUserBirth);

UserSex = readBinaryFile(fileUserSex);

UserName = readBinaryFile(fileUserName);

UserRank = SetUserLvl();

UserProgress = data.GetCurrentProgress();

}

catch (Exception^ e) {

MessageError->Show(e->Message, "Ошибка загрузки данных пользователя");

}

String^ pathToAvatarPng = User + "//avatar.png";

String^ pathToAvatarJpg = User + "//avatar.jpg";

if (File::Exists(fileUserSurname)) {

UserSurname = readBinaryFile(fileUserSurname);

}

if (File::Exists(pathToAvatarJpg)) {

pictureProfile->ImageLocation = pathToAvatarJpg;

pictureUserBar->ImageLocation = pathToAvatarJpg;

pictureProfileEdit->ImageLocation = pathToAvatarJpg;

}

else if (File::Exists(pathToAvatarPng)) {

pictureProfile->ImageLocation = pathToAvatarPng;

pictureUserBar->ImageLocation = pathToAvatarPng;

pictureProfileEdit->ImageLocation = pathToAvatarPng;

}

lblLogin->Text = User;

lblName->Text = "Имя: " + UserName;

lblSurname->Text = "Фамилия: " + UserSurname;

lblSex->Text = "Пол: " + UserSex;

lblBirth->Text = "Дата рождения: " + UserBirth;

lblEmail->Text = "Email: " + UserEmail;

lblRegDate->Text = "Дата регистрации: " + GetFolderCreationDate(User);

lblSexEdit->Text = "Пол:\n" + UserSex;

lblBirthEdit->Text = "Дата рождения:\n" + UserBirth;

lblProfileRank->Text = "Уровень: " + UserRank;

lblAnim1->Text = "Приветствуем вас на странице заданий, " + UserName + "!";

labelNameBar->Text = UserName;

labelRankBar->Text = UserRank;

StarsUser->Location = Point(labelRankBar->Right + 5, labelRankBar->Top);

textBoxUserName->Text = UserName;

textBoxUserSurname->Text = UserSurname;

textBoxEmail->Text = UserEmail;

}

void mainForm::DataChange() {

UserName = textBoxUserName->Text;

UserSurname = textBoxUserSurname->Text;

UserEmail = textBoxEmail->Text;

lblName->Text = "Имя: " + UserName;

lblSurname->Text = "Фамилия: " + UserSurname;

lblEmail->Text = "Email: " + UserEmail;

labelNameBar->Text = UserName;

}

void mainForm::DataSave() {

String^ fileUserEmail = User + "//userData.bin";

String^ fileUserName = User + "//userName.bin";

String^ fileUserSurname = User + "//userSurname.bin";

writeBinaryFile(fileUserEmail, UserEmail);

writeBinaryFile(fileUserName, UserName);

writeBinaryFile(fileUserSurname, UserSurname);

}

void mainForm::cfgLoad() {

Config^ config = config->LoadConfig();

borderlessForm->BorderRadius = config->borderForm;

borderlessForm->HasFormShadow = config->hasFormShadow;

dragMain->TransparentWhileDrag = config->dragTransparent;

greeting = config->greeting;

borderBtn = config->borderBtn;

volume = config->volume;

alwaysHideMenu = config->alwaysHideMenu;

borderForm = config->borderForm;

formShadow = config->hasFormShadow;

transparentWhileDrag = config->dragTransparent;

PythonWaveStyle = config->PythonWaveStyle;

TrackBorderBtn->Value = borderBtn;

TrackBorderForm->Value = borderForm;

TrackVolume->Value = volume;

labelBorderBtn->Text = Convert::ToString(borderBtn);

labelBorderForm->Text = Convert::ToString(borderForm);

labelVolume->Text = Convert::ToString(volume);

toggleAlwaysHide->Checked = alwaysHideMenu;

toggleGreeting->Checked = greeting;

toggleStyle->Checked = PythonWaveStyle;

toggleShadows->Checked = formShadow;

toggleTransparent->Checked = transparentWhileDrag;

RegisterMouseDownEvent(this, alwaysHideMenu);

PythonWaveStyleState(PythonWaveStyle);

}

void mainForm::cfgSave() {

Config^ config = gcnew Config();

config->alwaysHideMenu = alwaysHideMenu;

config->borderBtn = borderBtn;

config->borderForm = borderForm;

config->hasFormShadow = formShadow;

config->dragTransparent = transparentWhileDrag;

config->PythonWaveStyle = PythonWaveStyle;

config->greeting = greeting;

config->volume = volume;

config->SaveConfig();

}

String^ mainForm::SetUserLvl() {

String^ newRank;

String^ fileRankPath = User + "//lvl.bin";

if (!File::Exists(fileRankPath)) {

newRank = "Новичок";

writeBinaryFile(fileRankPath, newRank);

return newRank;

}

if (UserProgress >= 20 && UserProgress <= 39) {

newRank = "Исследователь";

StarsUser->Value = 1.5;

}

else if (UserProgress >= 40 && UserProgress <= 59) {

newRank = "Разработчик";

StarsUser->Value = 2.5;

}

else if (UserProgress >= 60 && UserProgress <= 79) {

newRank = "Инженер";

StarsUser->Value = 3.5;

}

else if (UserProgress >= 80 && UserProgress <= 100) {

newRank = "Мастер";

StarsUser->Value = 4.5;

}

else if (UserProgress >= 100) {

newRank = "Абсолют";

StarsUser->Value = 5;

StarsUser->RatingColor = Color::Red;

}

else {

newRank = "Новичок";

StarsUser->Value = 0;

return newRank;

}

String^ fileRank = User + "//lvl.bin";

writeBinaryFile(fileRank, newRank);

return newRank;

}

Void mainForm::labelRankBar\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

if (isDataLoaded) {

MessageInfo->Show("Вы получили новый уровень", "Поздравляем, " + UserRank);

}

}

Void mainForm::HideWAnimation(Forms::Control^ CONTROL) {

UI2::AnimatorNS::Animation^ MYANIM = gcnew UI2::AnimatorNS::Animation();

gunaTransition->HideSync(dynamic\_cast<Forms::Control^>(CONTROL), true, MYANIM->Transparent);

}

Void mainForm::ShowWAnimation(Forms::Control^ CONTROL) {

UI2::AnimatorNS::Animation^ MYANIM = gcnew UI2::AnimatorNS::Animation();

gunaTransition->ShowSync(dynamic\_cast<Forms::Control^>(CONTROL), true, MYANIM->Transparent);

}

«MainProfile.h»

#pragma once

#include "mainForm.h"

using namespace PythonWave;

int secondsLeftt = 30;

Void mainForm::buttonUploadImage\_Click(System::Object^ sender, System::EventArgs^ e) {

try {

openFileDialog1->Filter = "Image Files (\*.jpg;\*.png)|\*.jpg;\*.png";

openFileDialog1->FilterIndex = 1;

openFileDialog1->RestoreDirectory = true;

if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {

String^ filePath = openFileDialog1->FileName;

Drawing::Image^ image = System::Drawing::Image::FromFile(filePath);

pictureProfile->SizeMode = PictureBoxSizeMode::StretchImage;

pictureProfile->Image = image;

pictureProfileEdit->Image = image;

pictureUserBar->Image = image;

Drawing::Imaging::ImageFormat^ imageFormat = image->RawFormat;

String^ savePath = User + "\\avatar";

if (imageFormat->Equals(System::Drawing::Imaging::ImageFormat::Jpeg)) {

savePath += ".jpg";

image->Save(savePath, System::Drawing::Imaging::ImageFormat::Jpeg);

}

else if (imageFormat->Equals(System::Drawing::Imaging::ImageFormat::Png)) {

savePath += ".png";

image->Save(savePath, System::Drawing::Imaging::ImageFormat::Png);

}

pictureProfileEdit->Image->Save(savePath, imageFormat);

}

}

catch (Exception^ ex) {

MessageError->Show(ex->Message);

}

}

Void mainForm::buttonSendMail\_Click(System::Object^ sender, System::EventArgs^ e) {

if (textBoxEmail->Text == UserEmail) {

MessageWarning->Show("Этот Email уже привязан к вашему акканту");

return;

}

ClassMail^ EMAIL = gcnew ClassMail(this);

String^ userMail = UserEmail;

String^ userMailNew = Convert::ToString(textBoxEmail->Text);

securityCode = generateSecurityCode();

securityCodeNew = generateSecurityCode();

String^ mail = "Здравствуйте, ваш код для смены почты: " + Convert::ToString(securityCode);

String^ mailNew = "Здравствуйте, ваш код для подтверждения почты: " + Convert::ToString(securityCodeNew);

if (IsValidEmail(userMailNew)) {

if (EMAIL->SendEmail(userMail, "PythonWave: Смена почты", mail) && EMAIL->SendEmail(userMailNew, "PythonWave: Подтверждение новой почты", mailNew)) {

textBoxEmail->Enabled = false;

buttonSendMail->Enabled = false;

textBoxCode->Enabled = true;

textBoxCodeNew->Enabled = true;

buttonCheckCode->Enabled = true;

pictureBoxCheckMail->Visible = true;

labelTimer->Visible = true;

timerReMail->Start();

}

}

else {

MessageError->Show("Введите корректный адрес эл. почты", "Письмо не было отправлено");

textBoxEmail->BorderColor = Color::Red;

textBoxEmail->Clear();

}

}

Void mainForm::buttonCheckCode\_Click(System::Object^ sender, System::EventArgs^ e) {

try {

if (securityCode == Convert::ToInt32(textBoxCode->Text) && securityCodeNew == Convert::ToInt32(textBoxCodeNew->Text)) {

email\_confirmed = true;

textBoxCode->Enabled = false;

textBoxCodeNew->Enabled = false;

buttonCheckCode->Enabled = false;

pictureBoxCheckCode->Visible = true;

timerReMail->Stop();

labelTimer->Visible = false;

linkReMail->Visible = false;

MessageInfo->Show("Почта изменена", "Успешно");

}

else {

MessageError->Text = "Неверный код";

MessageError->Show();

}

}

catch (Exception^ e) {

MessageError->Show(e->Message);

}

}

Void mainForm::timerEmail\_Tick(System::Object^ sender, System::EventArgs^ e) {

secondsLeftt--;

if (secondsLeftt <= 0)

{

secondsLeftt = 30;

timerReMail->Stop();

labelTimer->Visible = false;

linkReMail->Visible = true;

linkReMail->Enabled = true;

}

else

{

labelTimer->Text = "Отправить повторно через " + secondsLeftt.ToString() + " секунд";

}

}

Void mainForm::linkReMail\_LinkClicked(System::Object^ sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e) {

enableMail(true);

email\_confirmed = false;

}

Void mainForm::btnChangePassword\_Click(System::Object^ sender, System::EventArgs^ e) {

try {

String^ filePath = User + "//data.bin";

size\_t storedHashPassword = readPassword(filePath);

if (storedHashPassword == 0) {

MessageError->Show("Не удалось прочитать старый пароль из файла");

return;

}

String^ enteredOldPass = textBoxPassOld->Text;

pin\_ptr<const wchar\_t> wchEntered = PtrToStringChars(enteredOldPass);

std::wstring wstrEntered(wchEntered);

std::string strEntered(wstrEntered.begin(), wstrEntered.end());

std::hash<std::string> hasher;

size\_t enteredHashPassword = hasher(strEntered);

if (enteredHashPassword != storedHashPassword) {

MessageError->Show("Неверный старый пароль");

return;

}

String^ newPass = textBoxPassNew->Text;

pin\_ptr<const wchar\_t> wchNewPass = PtrToStringChars(newPass);

std::wstring wstrNewPass(wchNewPass);

std::string strNewPass(wstrNewPass.begin(), wstrNewPass.end());

size\_t newHashPassword = hasher(strNewPass);

try {

FileStream^ fileStream = gcnew FileStream(filePath, FileMode::Create, FileAccess::Write);

BinaryWriter^ binaryWriter = gcnew BinaryWriter(fileStream);

binaryWriter->Write(static\_cast<UInt64>(newHashPassword));

binaryWriter->Close();

fileStream->Close();

}

catch (Exception^ e) {

MessageError->Show("Ошибка записи файла: " + e->Message);

}

MessageInfo->Show("Пароль успешно изменен", "Поздравляем");

if (Directory::Exists("logs")) {

try {

Directory::Delete("logs", true);

}

catch (Exception^ e) {

MessageBox::Show(e->Message);

}

}

}

catch (Exception^ e) {

MessageError->Show(e->Message);

}

}

Void mainForm::linkREMOVEACC\_LinkClicked(System::Object^ sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e) {

if (textBoxPassOld->Text == "") {

MessageInfo->Show("Введите ваш пароль в поле 'Старый пароль'");

return;

}

String^ enteredOldPass = textBoxPassOld->Text;

pin\_ptr<const wchar\_t> wchEntered = PtrToStringChars(enteredOldPass);

std::wstring wstrEntered(wchEntered);

std::string strEntered(wstrEntered.begin(), wstrEntered.end());

String^ filePath = User + "//data.bin";

std::hash<std::string> hasher;

size\_t enteredHashPassword = hasher(strEntered);

size\_t storedHashPassword = readPassword(filePath);

if (enteredHashPassword != storedHashPassword) {

MessageError->Show("Неверный пароль");

return;

}

Forms::DialogResult result = MessageQuestion->Show("Ваш аккаунт будет удален", "Вы уверены?");

if (result == Forms::DialogResult::Yes) {

DeleteDirectory(User);

MessageDefault->Show("До скорых встреч, " + UserName + "!", "Аккаунт удален");

if (Directory::Exists("logs")) {

try {

Directory::Delete("logs", true);

}

catch (Exception^ e) {

MessageBox::Show(e->Message);

}

}

ClassFade^ Fade = gcnew ClassFade(this);

Fade->SetAnimation("close");

}

}

Void mainForm::buttonResume\_Click(System::Object^ sender, System::EventArgs^ e) {

Pages->SelectTab(pageProfileEdit);

}

Void mainForm::btnCancelChanges\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CheckSave())

Pages->SelectTab(pageProfile);

}

Void mainForm::btnProfileSave\_Click(System::Object^ sender, System::EventArgs^ e) {

if (textBoxUserName->TextLength < 3 || textBoxUserSurname->TextLength < 3) {

if (textBoxUserName->TextLength < 3)

textBoxUserName->BorderColor = Color::Red;

if (textBoxUserSurname->TextLength < 3)

textBoxUserSurname->BorderColor = Color::Red;

MessageWarning->Show("Введите полное имя и фамилию");

return;

}

if (!email\_confirmed) {

MessageWarning->Show("Вы ввели новый Email, подтвердите его или введите старый");

textBoxEmail->BorderColor = Color::Red;

return;

}

isProfileSaved = true;

DataChange();

DataSave();

MessageInfo->Show("Сохранено", "Успешно");

}

Boolean mainForm::CheckProfileSave(UI2::WinForms::Guna2TextBox^ TB, String^ field) {

if (TB->Text != field) {

isProfileSaved = false;

return false;

}

return true;

}

Void mainForm::textBoxUserName\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

if (textBoxUserName->Text != UserName) {

isProfileSaved = false;

}

}

Void mainForm::textBoxUserSurname\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

CheckProfileSave(textBoxUserSurname, UserSurname);

}

Void mainForm::textBoxEmail\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

email\_confirmed = CheckProfileSave(textBoxEmail, UserEmail);

}

«MainTasks.h»

#pragma once

#include "mainForm.h"

#include "MyPython.h"

#include "ClassTasks.h"

using namespace PythonWave;

using namespace System;

int currentAnim = 1;

double animSeconds = 2;

int secondsToStartAnim = 4;

bool canSaveFunc = false;

void mainForm::SyntaxHighlight(RichTextBox^ richTB) {

int selectionStart = richTB->SelectionStart;

int selectionLength = richTB->SelectionLength;

array<String^>^ keywords = { "False", "None", "True", "and", "as", "assert", "async", "await", "break", "class", "continue", "def", "del", "elif", "else", "except", "finally", "for", "from", "global", "if", "import", "in", "is", "lambda", "nonlocal", "not", "or", "pass", "raise", "return", "try", "while", "with", "yield", "len", "sum" };

Color defaultTextColor = Color::FromArgb(238, 238, 238);

Color keywordColor = Color::Violet;

richTB->SelectAll();

richTB->SelectionColor = defaultTextColor;

richTB->SelectionFont = gcnew System::Drawing::Font(richTB->Font, FontStyle::Regular);

std::string text = msclr::interop::marshal\_as<std::string>(richTB->Text);

for each (String ^ keyword in keywords) {

std::string keyword\_pattern = msclr::interop::marshal\_as<std::string>(keyword);

std::regex word\_regex("\\b" + keyword\_pattern + "\\b");

std::sregex\_iterator words\_begin(text.begin(), text.end(), word\_regex);

std::sregex\_iterator words\_end;

while (words\_begin != words\_end) {

std::smatch match = \*words\_begin;

int start = match.position();

int length = match.length();

richTB->Select(start, length);

richTB->SelectionColor = keywordColor;

richTB->SelectionFont = gcnew System::Drawing::Font(richTB->Font, FontStyle::Bold);

++words\_begin;

}

}

richTB->Select(selectionStart, selectionLength);

richTB->SelectionColor = defaultTextColor;

}

Void mainForm::btnCourses\_Click(System::Object^ sender, System::EventArgs^ e) {

if (isCoursesVisited)

funcSelectTab(pageTasks);

else if (!courseAnimationState)

{

currentAnim = 1;

animSeconds = 3;

Pages->AllowTransitions = true;

funcSelectTab(anim1);

timerAnim->Start();

courseAnimationState = true;

if (menu) {

menu = false;

timerMenu->Start();

}

}

}

Void mainForm::btnSync\_Click(System::Object^ sender, System::EventArgs^ e) {

Pages->SelectTab(anim4);

currentAnim = 4;

animSeconds = 3;

timerAnim->Start();

}

Void mainForm::btnSync1\_Click(System::Object^ sender, System::EventArgs^ e) {

funcSelectTab(pageTasks);

animSeconds = 3;

}

Void mainForm::timerAnim\_Tick(System::Object^ sender, System::EventArgs^ e) {

switch (currentAnim)

{

case 1:

--animSeconds;

if (animSeconds == 0) {

Pages->SelectTab(anim2);

currentAnim = 2;

animSeconds = 3;

}

break;

case 2:

--animSeconds;

if (animSeconds == 0) {

Pages->SelectTab(anim3);

currentAnim = 3;

animSeconds = 3;

}

break;

case 3:

--animSeconds;

if (animSeconds == 0) {

if (lblSync1->Visible == false) {

ShowWAnimation(lblSync1);

animSeconds = 1;

break;

}

if (lblSync2->Visible == false) {

ShowWAnimation(lblSync2);

animSeconds = 1;

break;

}

if (lblSync3->Visible == false) {

ShowWAnimation(lblSync3);

animSeconds = 1;

break;

}

if (lblSync4->Visible == false) {

ShowWAnimation(lblSync4);

animSeconds = 1;

break;

}

if (lblSync5->Visible == false) {

ShowWAnimation(lblSync5);

animSeconds = 3;

break;

}

if (btnSync->Visible == false) {

ShowWAnimation(btnSync);

timerAnim->Stop();

break;

}

}

break;

case 4:

--animSeconds;

if (animSeconds == 0) {

currentAnim = 5;

ShowWAnimation(btnSync1);

}

break;

case 5:

Pages->AllowTransitions = false;

isCoursesVisited = true;

courseAnimationState = false;

writeBinaryFile("script//logs.bin", Convert::ToString(isCoursesVisited));

timerAnim->Stop();

break;

default:

break;

}

}

Void mainForm::richTask1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

SyntaxHighlight(richTask);

canSaveFunc = false;

}

Void mainForm::TaskText\_TextChanged\_1(System::Object^ sender, System::EventArgs^ e) {

TaskText->Multiline = true;

TaskText->WordWrap = true;

TaskText->ScrollBars = ScrollBars::Vertical;

}

Void mainForm::LoadTask() {

try {

String^ pathToText = "script//text//" + CurrentDifficulty + "//" + CurrentTask + ".txt";

String^ pathToFunc = "script//text//" + CurrentDifficulty + "//" + CurrentTask + ".py";

FileStream^ fsText = gcnew FileStream(pathToText, FileMode::Open, FileAccess::Read);

StreamReader^ srText = gcnew StreamReader(fsText);

String^ text = srText->ReadToEnd();

srText->Close();

fsText->Close();

FileStream^ fsFunc = gcnew FileStream(pathToFunc, FileMode::Open, FileAccess::Read);

StreamReader^ srFunc = gcnew StreamReader(fsFunc);

String^ func = srFunc->ReadToEnd();

srFunc->Close();

fsFunc->Close();

TaskText->Text = text;

richTask->Text = func;

SyntaxHighlight(richTask);

PagesTasks->SelectTab(TaskPage);

}

catch (Exception^ e) {

MessageBox::Show(e->Message);

}

}

void mainForm::InitializeTaskArrays() {

easyTasks = gcnew array<String^> {"add", "subtract", "multiply", "divide", "even\_or\_odd"};

middleTasks = gcnew array<String^> {"positive\_sum", "better\_than\_average", "reverse\_seq"};

hardTasks = gcnew array<String^> {"get\_count", "high\_and\_low", "square\_digits"};

veryHardTasks = gcnew array<String^> {"get\_char", "get\_middle", "symmetric\_point"};

}

array<String^>^ mainForm::GetTaskArray() {

if (CurrentDifficulty == "easy") {

return easyTasks;

}

else if (CurrentDifficulty == "middle") {

return middleTasks;

}

else if (CurrentDifficulty == "hard") {

return hardTasks;

}

else if (CurrentDifficulty == "very\_hard") {

return veryHardTasks;

}

return nullptr;

}

Void mainForm::NextTask() {

InitializeTaskArrays();

array<String^>^ tasks = GetTaskArray();

if (tasks == nullptr) {

MessageError->Show("Error: Invalid difficulty", "Transition is not possible");

return;

}

TaskIndex = Array::IndexOf(tasks, CurrentTask);

if (TaskIndex == -1) {

MessageError->Show("Error: Invalid Index", "Transition is not possible");

return;

}

TaskIndex = (TaskIndex + 1) % tasks->Length;

CurrentTask = tasks[TaskIndex];

LoadTask();

}

Void mainForm::PreviousTask() {

InitializeTaskArrays();

array<String^>^ tasks = GetTaskArray();

if (tasks == nullptr) {

MessageError->Show("Error: Invalid difficulty", "Transition is not possible");

return;

}

TaskIndex = Array::IndexOf(tasks, CurrentTask);

if (TaskIndex == -1) {

MessageError->Show("Error: Invalid Index", "Transition is not possible");

return;

}

TaskIndex = (TaskIndex - 1 + tasks->Length) % tasks->Length;

CurrentTask = tasks[TaskIndex];

LoadTask();

}

Void mainForm::btnPerviousTask\_Click(System::Object^ sender, System::EventArgs^ e) {

PreviousTask();

}

Void mainForm::btnNextTask\_Click(System::Object^ sender, System::EventArgs^ e) {

NextTask();

}

Void mainForm::btnBack\_Click(System::Object^ sender, System::EventArgs^ e) {

PagesTasks->SelectTab(TasksMain);

}

Void mainForm::lblPanelTaskInfo\_Click(System::Object^ sender, System::EventArgs^ e) {

lblPanelTaskInfo->Visible = false;

}

Void mainForm::panelTask\_Click(System::Object^ sender, System::EventArgs^ e) {

lblPanelTaskInfo->Visible = false;

}

void mainForm::SetTaskAndLoad(String^ task, String^ difficulty) {

CurrentTask = task;

CurrentDifficulty = difficulty;

LoadTask();

}

Void mainForm::btnTaskAdd\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("add", "easy");

}

Void mainForm::btnTaskDivide\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("divide", "easy");

}

Void mainForm::btnTaskMultiply\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("multiply", "easy");

}

Void mainForm::btnTaskSubtract\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("subtract", "easy");

}

Void mainForm::btnTaskEvenOrOdd\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("even\_or\_odd", "easy");

}

Void mainForm::btnTaskPositiveSum\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("positive\_sum", "middle");

}

Void mainForm::btnTaskBetterThanAverage\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("better\_than\_average", "middle");

}

Void mainForm::btnReverseSeq\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("reverse\_seq", "middle");

}

Void mainForm::btnTaskGetCount\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("get\_count", "hard");

}

Void mainForm::btnTaskHighAndLow\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("high\_and\_low", "hard");

}

Void mainForm::btnTaskSquareDigits\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("square\_digits", "hard");

}

Void mainForm::btnTaskGetChar\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("get\_char", "very\_hard");

}

Void mainForm::btnHero\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("get\_middle", "very\_hard");

}

Void mainForm::btnSymmetricPoint\_Click(System::Object^ sender, System::EventArgs^ e) {

SetTaskAndLoad("symmetric\_point", "very\_hard");

}

void mainForm::PyRun(String^ code) {

if (String::IsNullOrEmpty(CurrentTask)) {

MessageError->Show("CurrentTask is Null or Empty", "Ошибка переменной: String^ CurrentTask");

return;

}

MyPython PyRunner;

String^ PythonOutput = PyRunner.Start(code);

String^ pathToResult = "script//result.txt";

if (!File::Exists(pathToResult)) {

return;

}

FileStream^ fs = gcnew FileStream(pathToResult, FileMode::Open, FileAccess::Read);

StreamReader^ sr = gcnew StreamReader(fs);

String^ result = sr->ReadToEnd()->Trim();

sr->Close();

fs->Close();

if (result == "OK") {

canSaveFunc = true;

ClassTasks^ validate = gcnew ClassTasks(User);

bool isTaskCompleted = validate->GetTaskValue(CurrentTask);

if (isTaskCompleted) {

MessageInfo->Show("Задача решена верно.", "Поздравляем!");

MessageWarning->Show("Вы не получите баллов за это решение, так как вы уже решили эту задачу ранее.");

return;

}

ClassProgress^ progress = gcnew ClassProgress(User);

int points;

if (CurrentDifficulty == "easy") {

progress->SolveTaskB();

points = progress->getTaskPoint(1);

validate->completedB\_count++;

}

else if (CurrentDifficulty == "middle") {

progress->SolveTaskA();

points = progress->getTaskPoint(2);

validate->completedA\_count++;

}

else if (CurrentDifficulty == "hard") {

progress->SolveTaskS();

points = progress->getTaskPoint(3);

validate->completedS\_count++;

}

else if (CurrentDifficulty == "very\_hard") {

progress->SolveTaskSplus();

points = progress->getTaskPoint(4);

validate->completedSplus\_count++;

}

else {

MessageError->Show("CurrentDifficulty не определена", "Ошибка PyRun()");

return;

}

validate->SolveTask(CurrentTask);

validate->SetTaskCompletionDate(CurrentTask);

String^ text = "баллов";

if (points == 1)

text = "балл";

else if (points > 1 && points < 5)

text = "балла";

else if (points > 4)

text = "баллов";

MessageInfo->Show("Задача решена верно. Вы получите " + points + " " + text, "Поздравляем!");

UserProgress = progress->GetCurrentProgress();

}

else if (result == "ERROR") {

MessageError->Show("Ошибка в коде", "Внимание");

}

else if (result == "FAILED") {

MessageWarning->Show("Задача решена неверно", "PythonWave");

}

else {

MessageBox::Show(result, "Неопределенный результат");

}

if (result == "ERROR" || result == "FAILED") {

try {

String^ trimmedResult = PythonOutput->Substring(PythonOutput->IndexOf("Traceback"));

MessageBox::Show(trimmedResult, "Вывод Python");

}

catch (Exception^ e) {

MessageBox::Show(PythonOutput, "Вывод Python");

}

}

File::Delete(pathToResult);

}

int mainForm::GetTasksCompletedCount(int substract) {

ClassTasks date(User);

DateTime end = DateTime::Now;

DateTime start = end.AddDays(substract);

return date.getCompletedCount(start, end);

}

Void mainForm::btnTestCode\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ code = richTask->Text;

PyRun(code);

}

Void mainForm::btnSaveCode\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ pathToFunc = "script/text/" + CurrentDifficulty + "/" + CurrentTask + ".py";

String^ code = richTask->Text;

if (canSaveFunc) {

try {

FileStream^ fs = gcnew FileStream(pathToFunc, FileMode::Create, FileAccess::Write);

StreamWriter^ sw = gcnew StreamWriter(fs);

sw->Write(code);

sw->Close();

fs->Close();

MessageInfo->Show("Ваше решение сохранено", "Успешно");

}

catch (Exception^ e) {

MessageError->Show(e->Message, "Ошибка сохранения");

}

}

else {

MessageWarning->Show("Для сохранения вам нужно успешно протестировать код, затем сразу сохранить его.", "Внимание");

}

}

Void mainForm::btnReport\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ period = dropdownReport->Text;

ClassTasks^ data = gcnew ClassTasks(User);

String^ taskStates = data->GetTaskStates();

int completed = 0;

if (period == "Период") {

MessageWarning->Show("Для получения отчета выберите период", "Внимание");

return;

}

DateTime endDate = DateTime::Now;

DateTime startDate;

if (period == "1 Неделя") {

startDate = endDate.AddDays(-7);

}

else if (period == "2 Недели") {

startDate = endDate.AddDays(-14);

}

else if (period == "3 Недели") {

startDate = endDate.AddDays(-21);

}

else if (period == "4 Недели") {

startDate = endDate.AddDays(-28);

}

else if (period == "Месяц") {

startDate = endDate.AddMonths(-1);

}

else {

period = "1 Неделя";

startDate = endDate.AddDays(-7);

}

completed = data->getCompletedCount(startDate, endDate);

String^ MaxDiff = data->getMaxDifficulty();

array<String^>^ EzTasks = { "add", "multiply", "divide", "subtract", "even\_or\_odd" };

array<String^>^ MidTasks = { "better\_than\_average", "positive\_sum", "reverse\_seq" };

array<String^>^ HardTasks = { "get\_count", "high\_and\_low", "square\_digits" };

array<String^>^ VeryHardTasks = { "get\_char", "symmetric\_point", "get\_middle" };

int ezCount = 0, midCount = 0, hardCount = 0, veryHardCount = 0;

String^ ezTasksCompleted = "";

String^ midTasksCompleted = "";

String^ hardTasksCompleted = "";

String^ veryHardTasksCompleted = "";

for each (KeyValuePair<String^, bool> kvp in data->taskDict) {

if (kvp.Value) {

if (Array::IndexOf(EzTasks, kvp.Key) != -1) {

ezCount++;

ezTasksCompleted += kvp.Key + ", ";

}

else if (Array::IndexOf(MidTasks, kvp.Key) != -1) {

midCount++;

midTasksCompleted += kvp.Key + ", ";

}

else if (Array::IndexOf(HardTasks, kvp.Key) != -1) {

hardCount++;

hardTasksCompleted += kvp.Key + ", ";

}

else if (Array::IndexOf(VeryHardTasks, kvp.Key) != -1) {

veryHardCount++;

veryHardTasksCompleted += kvp.Key + ", ";

}

}

}

if (ezTasksCompleted->Length > 0) ezTasksCompleted = ezTasksCompleted->Substring(0, ezTasksCompleted->Length - 2);

if (midTasksCompleted->Length > 0) midTasksCompleted = midTasksCompleted->Substring(0, midTasksCompleted->Length - 2);

if (hardTasksCompleted->Length > 0) hardTasksCompleted = hardTasksCompleted->Substring(0, hardTasksCompleted->Length - 2);

if (veryHardTasksCompleted->Length > 0) veryHardTasksCompleted = veryHardTasksCompleted->Substring(0, veryHardTasksCompleted->Length - 2);

String^ FullMessage = "Количество решенных задач: " + completed.ToString() + "\n\n";

FullMessage += "Легкие задачи:\nКол-во: " + ezCount + "\nРешенные задачи: " + ezTasksCompleted + "\n\n";

FullMessage += "Средние задачи:\nКол-во: " + midCount + "\nРешенные задачи: " + midTasksCompleted + "\n\n";

FullMessage += "Сложные задачи:\nКол-во: " + hardCount + "\nРешенные задачи: " + hardTasksCompleted + "\n\n";

FullMessage += "Очень сложные задачи:\nКол-во: " + veryHardCount + "\nРешенные задачи: " + veryHardTasksCompleted + "\n\n";

FullMessage += "Желаете экспортировать отчет?";

String^ caption = "Отчет за период: " + period;

if (MessageQuestion->Show(FullMessage, caption) == ::DialogResult::Yes) {

SaveReportToFile(FullMessage);

}

}

Void mainForm::SaveReportToFile(String^ reportContent) {

SaveFileDialog^ saveFileDialog = gcnew SaveFileDialog();

saveFileDialog->Filter = "Text File|\*.txt";

saveFileDialog->Title = "Сохранить отчет";

String^ defaultFileName = "Отчет " + DateTime::Now.ToString("dd.MM") + ".txt";

saveFileDialog->FileName = defaultFileName;

if (saveFileDialog->ShowDialog() == ::DialogResult::OK && saveFileDialog->FileName != "") {

try {

StreamWriter^ sw = gcnew StreamWriter(saveFileDialog->FileName);

int exportQuestionIndex = reportContent->LastIndexOf("Желаете экспортировать отчет?");

if (exportQuestionIndex != -1) {

reportContent = reportContent->Substring(0, exportQuestionIndex);

}

sw->Write(reportContent);

sw->WriteLine();

sw->WriteLine("Отчет создан: " + DateTime::Now.ToString("dd.MM.yyyy HH:mm:ss"));

sw->WriteLine("Делитесь своими успехами с окружающими! С уважением PythonWave.");

sw->Close();

MessageBox::Show("Отчет успешно сохранен!", "Успех", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

catch (Exception^ ex) {

MessageError->Show(ex->Message);

}

}

}

«MainSettings.h»

#pragma once

#include "mainForm.h"

Void mainForm::TrackBorderForm\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e) {

borderlessForm->BorderRadius = TrackBorderForm->Value;

borderForm = TrackBorderForm->Value;

labelBorderForm->Text = Convert::ToString(borderForm);

}

Void mainForm::TrackBorderBtn\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e) {

borderBtn = TrackBorderBtn->Value;

btnSettingsCancel->BorderRadius = TrackBorderBtn->Value;

btnSettingsSave->BorderRadius = TrackBorderBtn->Value;

labelBorderBtn->Text = Convert::ToString(TrackBorderBtn->Value);

}

Void mainForm::TrackVolume\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e) {

volume = TrackVolume->Value;

labelVolume->Text = Convert::ToString(volume);

}

Void mainForm::toggleTransparent\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

transparentWhileDrag = toggleTransparent->Checked;

dragMain->TransparentWhileDrag = transparentWhileDrag;

}

Void mainForm::toggleGreeting\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

greeting = toggleGreeting->Checked;

}

Void mainForm::toggleShadows\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

formShadow = toggleShadows->Checked;

}

Void mainForm::toggleAlwaysHide\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

alwaysHideMenu = toggleAlwaysHide->Checked;

RegisterMouseDownEvent(this, alwaysHideMenu);

}

Void mainForm::btnMessageShadow\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageWarning->Show("Эта функция на стадии ранней разработки");

}

Void mainForm::btnSettingsSave\_Click(System::Object^ sender, System::EventArgs^ e) {

cfgSave();

MessageInfo->Show("Настройки успешно сохранены");

}

Void mainForm::toggleStyle\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

PythonWaveStyle = toggleStyle->Checked;

PythonWaveStyleState(PythonWaveStyle);

}

Void mainForm::PythonWaveStyleState(bool enable) {

if (enable) {

this->FormBorderStyle = Forms::FormBorderStyle::None;

Config^ tempCfg = tempCfg->LoadConfig();

borderForm = tempCfg->borderForm;

tempCfg = nullptr;

}

else {

this->FormBorderStyle = Forms::FormBorderStyle::FixedToolWindow;

borderForm = 0;

}

btnExit->Visible = enable;

btnMinimize->Visible = enable;

TrackBorderForm->Enabled = enable;

borderlessForm->BorderRadius = borderForm;

TrackBorderForm->Value = borderForm;

labelBorderForm->Text = Convert::ToString(borderForm);

}

void mainForm::RegisterMouseDownEvent(Control^ parent, bool enable)

{

if (enable && parent->Name != "btnMenu")

{

parent->MouseDown += mouseDownHandler;

}

else

{

parent->MouseDown -= mouseDownHandler;

}

for each (Control ^ child in parent->Controls)

{

RegisterMouseDownEvent(child, enable);

}

}

Void mainForm::textBoxUserName\_Click(System::Object^ sender, System::EventArgs^ e) {

textBoxUserName->BorderColor = Color::Silver;

}

Void mainForm::textBoxUserSurname\_Click(System::Object^ sender, System::EventArgs^ e) {

textBoxUserSurname->BorderColor = Color::Silver;

}

Void mainForm::textBoxEmail\_Click(System::Object^ sender, System::EventArgs^ e) {

textBoxEmail->BorderColor = Color::Silver;

}

«MainHome.h»

#pragma once

#include "mainForm.h"

#include "ClassTasks.h"

using namespace PythonWave;

using namespace System;

String^ mainForm::GetGreetingBasedOnTime() {

DateTime now = DateTime::Now;

int hour = now.Hour;

if (hour >= 5 && hour < 12) {

return "Доброе утро, " + UserName + "!";

}

else if (hour >= 12 && hour < 17) {

return "Добрый день, " + UserName;

}

else if (hour >= 17 && hour < 22) {

return "Добрый вечер, " + UserName;

}

else {

return "Доброй ночи, " + UserName;

}

}

String^ GetDayOfWeek() {

DateTime now = DateTime::Now;

DayOfWeek dayOfWeek = now.DayOfWeek;

String^ dayName;

switch (dayOfWeek)

{

case DayOfWeek::Monday:

dayName = "Пн";

break;

case DayOfWeek::Tuesday:

dayName = "Вт";

break;

case DayOfWeek::Wednesday:

dayName = "Ср";

break;

case DayOfWeek::Thursday:

dayName = "Чт";

break;

case DayOfWeek::Friday:

dayName = "Пт";

break;

case DayOfWeek::Saturday:

dayName = "Сб";

break;

case DayOfWeek::Sunday:

dayName = "Вс";

break;

}

return dayName;

}

void mainForm::InitializePhrazes() {

phrases = gcnew array<String^> {

"Каждая строка кода приближает тебя к успеху.",

"Сегодня ты можешь создать будущее.",

"Кодь так, будто мир зависит от твоей программы.",

"Ни один баг не устоит перед твоим упорством.",

"Код — это искусство, а ты — художник.",

"Не бойся ошибок, они — твои лучшие учителя.",

"Каждый день — новая возможность стать лучше.",

"Сегодня ты можешь написать лучший код в своей жизни.",

"Не останавливайся на достигнутом, всегда стремись к большему.",

"Кодирование — это путешествие, наслаждайся каждым шагом.",

"Всё невозможное возможно для того, кто пытается.",

"Твои идеи могут изменить мир, просто начни кодить.",

"Каждый байт твоего кода имеет значение.",

"Сегодня ты решишь проблему, которая казалась невозможной.",

"Пиши код, который будет вдохновлять других.",

"Каждая задача — это шанс проявить себя.",

"Твои навыки растут с каждым написанным символом.",

"Код — это магия, а ты — волшебник.",

"Не сдавайся, великое всегда требует времени.",

"Сегодня — идеальный день для создания чего-то нового.",

"Каждая ошибка — это шаг к совершенству.",

"Кодинг — это твоя суперсила.",

"Твоя решимость сильнее любых препятствий.",

"Кодирование — это твоя возможность создать что-то уникальное.",

"Смелость и настойчивость — твои лучшие инструменты.",

"Пусть сегодня будет день великих достижений.",

"Каждая строка кода приближает тебя к цели.",

"Твоя работа важна, твои усилия ценны.",

"Сегодня ты сделаешь то, что не удавалось вчера.",

"Ты способен на всё, просто начни кодить.",

"Код — это твоя возможность выразить свои мысли.",

"Создавай код, который будет работать на тебя.",

"Каждый новый день — это шанс начать заново.",

"Твои усилия сегодня приведут к результатам завтра.",

"Пиши код, который будет вдохновлять будущие поколения."

};

}

String^ mainForm::GetPhraseForToday() {

int day = DateTime::Now.Day;

int index = (day - 1) % phrases->Length;

return phrases[index];

}

void mainForm::LoadHomePage() {

InitializePhrazes();

lblHello->Text = GetGreetingBasedOnTime();

lblSovet->Text = GetPhraseForToday();

SetCenter(pageHome, lblHello, 1);

SetCenter(pnlHome2, lblPnlHome2, 1);

SetCenter(pageHome, lblHello, 1);

SetCenter(pnlHome1n1, btnRead, 1);

SetCenter(pnlHome1, btnReadBook, 1);

ReadTimeFromFile();

InitializePanelBook();

ClassTasks tasks(User);

currentDayOfWeek = GetDayOfWeek();

array<String^>^ DaysOfWeek = gcnew array<String^>(7);

DaysOfWeek[0] = "Пн";

DaysOfWeek[1] = "Вт";

DaysOfWeek[2] = "Ср";

DaysOfWeek[3] = "Чт";

DaysOfWeek[4] = "Пт";

DaysOfWeek[5] = "Сб";

DaysOfWeek[6] = "Вс";

TasksDataset->DataPoints->Clear();

DateTime today = DateTime::Now.Date;

DateTime startOfWeek = today.AddDays(-(int)today.DayOfWeek + (int)DayOfWeek::Monday);

for (int i = 0; i < DaysOfWeek->Length; ++i) {

DateTime dayToCheck = startOfWeek.AddDays(i);

int completedTasks = tasks.getCompletedCount(dayToCheck, dayToCheck.AddDays(1));

TasksDataset->DataPoints->Add(DaysOfWeek[i], completedTasks);

}

circleProgress->Value = UserProgress;

}

void mainForm::InitializePanelBook() {

String^ filePath = User + "//book//lastpage.txt";

String^ pageName = "none";

if (File::Exists(filePath)) {

pageName = File::ReadAllText(filePath);

int tabPageIndex = -1;

for (int i = 0; i < Book->TabCount; ++i) {

if (Book->TabPages[i]->Text == pageName) {

tabPageIndex = i;

break;

}

}

if (tabPageIndex != -1) {

Book->SelectTab(tabPageIndex);

dropdownPages->SelectedIndex = tabPageIndex;

String^ selectedPageText = dropdownPages->Items[tabPageIndex]->ToString();

StringBuilder^ newText = gcnew StringBuilder();

for each (Char c in selectedPageText)

{

if (!Char::IsDigit(c) && c != '.')

{

newText->Append(c);

}

}

String^ resultText = newText->ToString();

lblLastPage->Text = resultText;

pnlHome1n1->Visible = true;

}

else {

MessageWarning->Show("tabPageIndex: " + tabPageIndex.ToString(), "Ошибка индекса");

pageName = "none";

}

}

if (pageName == "none") {

pnlHome1->Visible = true;

return;

}

}

Void mainForm::btnReadBook\_Click(System::Object^ sender, System::EventArgs^ e) {

funcSelectTab(pageBook);

btnBook->Checked = true;

}

Void mainForm::btnRead\_Click(System::Object^ sender, System::EventArgs^ e) {

funcSelectTab(pageBook);

btnBook->Checked = true;

}

void mainForm::ReadTimeFromFile()

{

String^ pathToTimer = User + "//time.bin";

try

{

if (File::Exists(pathToTimer))

{

FileStream^ fs = gcnew FileStream(pathToTimer, FileMode::Open, FileAccess::Read);

BinaryReader^ reader = gcnew BinaryReader(fs);

String^ line = reader->ReadString();

secondsSpent = Int32::Parse(line);

reader->Close();

}

else

{

secondsSpent = 0;

}

}

catch (Exception^ ex)

{

MessageBox::Show("Error reading time from file: " + ex->Message);

secondsSpent = 0;

}

double hoursSpent = secondsSpent / 3600.0;

lblTimerInApp->Text = hoursSpent.ToString("0.0") + "ч";

}

void mainForm::WriteTimeToFile()

{

String^ pathToTimer = User + "//time.bin";

try

{

FileStream^ fs = gcnew FileStream(pathToTimer, FileMode::OpenOrCreate, FileAccess::Write);

BinaryWriter^ writer = gcnew BinaryWriter(fs);

writer->Write(secondsSpent.ToString());

writer->Close();

}

catch (Exception^ ex)

{

MessageBox::Show("Error writing time to file: " + ex->Message);

}

}

«mainForm.h»

#pragma once

#include "config.h"

#include "ClassFade.h"

#include "ClassProgress.h"

#include "ClassTasks.h"

#include <regex>

#include <string>

#include <msclr\marshal\_cppstd.h>

namespace PythonWave {

using namespace Guna;

using namespace System;

using namespace System::IO;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::Drawing;

using namespace System::Diagnostics;

using namespace System::Collections;

using namespace System::ComponentModel;

using namespace System::Windows::Forms;

using namespace System::Collections::Generic;

public ref class mainForm : public Form

{

private:

System::Windows::Forms::Timer^ timerAnim;

Bunifu::UI::WinForms::BunifuPages^ Pages;

System::Windows::Forms::TabPage^ pageBook;

Bunifu::UI::WinForms::BunifuPages^ Book;

System::Windows::Forms::TabPage^ pgBookStart;

System::Windows::Forms::RichTextBox^ richStart;

System::Windows::Forms::TabPage^ Book1;

System::Windows::Forms::RichTextBox^ richBook1;

System::Windows::Forms::TabPage^ Book2;

System::Windows::Forms::RichTextBox^ richBook2;

System::Windows::Forms::TabPage^ Book3;

System::Windows::Forms::RichTextBox^ richBook3;

System::Windows::Forms::TabPage^ Book4;

System::Windows::Forms::RichTextBox^ richBook4;

System::Windows::Forms::TabPage^ Book5;

System::Windows::Forms::RichTextBox^ richBook5;

System::Windows::Forms::TabPage^ Book6;

System::Windows::Forms::RichTextBox^ richBook6;

System::Windows::Forms::TabPage^ Book7;

System::Windows::Forms::RichTextBox^ richBook7;

System::Windows::Forms::TabPage^ Book8;

System::Windows::Forms::RichTextBox^ richBook8;

System::Windows::Forms::TabPage^ Book9;

System::Windows::Forms::RichTextBox^ richBook9;

System::Windows::Forms::TabPage^ Book10;

System::Windows::Forms::RichTextBox^ richBook10;

System::Windows::Forms::TabPage^ Book11;

System::Windows::Forms::RichTextBox^ richBook11;

System::Windows::Forms::TabPage^ Book12;

System::Windows::Forms::RichTextBox^ richBook12;

System::Windows::Forms::TabPage^ Book13;

System::Windows::Forms::RichTextBox^ richBook13;

System::Windows::Forms::TabPage^ Book14;

System::Windows::Forms::RichTextBox^ richTextBox1;

System::Windows::Forms::Panel^ myToolbox;

Guna::UI2::WinForms::Guna2ComboBox^ dropdownPages;

Guna::UI2::WinForms::Guna2Button^ btnPreviousBook;

Guna::UI2::WinForms::Guna2Button^ btnBookNext;

Guna::UI2::WinForms::Guna2Button^ btnItalic;

Guna::UI2::WinForms::Guna2Button^ btnStrikeOut;

Guna::UI2::WinForms::Guna2Button^ btnClearFilters;

Guna::UI2::WinForms::Guna2Button^ btnThick;

Guna::UI2::WinForms::Guna2Button^ btnUnderline;

Guna::UI2::WinForms::Guna2CircleButton^ btnQuestionBook;

Guna::UI2::WinForms::Guna2Button^ btnHighlight;

System::Windows::Forms::TabPage^ pageTasks;

Guna::UI2::WinForms::Guna2Panel^ panelTasks;

System::Windows::Forms::TabPage^ pageProfile;

System::Windows::Forms::Panel^ panelProfileData;

Guna::UI2::WinForms::Guna2Button^ btnProfileEdit;

System::Windows::Forms::Label^ lblProfileRank;

System::Windows::Forms::Label^ lblEmail;

System::Windows::Forms::Label^ lblRegDate;

System::Windows::Forms::Label^ lblBirth;

System::Windows::Forms::Label^ lblSex;

System::Windows::Forms::Label^ lblSurname;

System::Windows::Forms::Label^ lblName;

System::Windows::Forms::Label^ lblLogin;

System::Windows::Forms::PictureBox^ pictureProfile;

System::Windows::Forms::TabPage^ pageStats;

System::Windows::Forms::TabPage^ pageSettings;

System::Windows::Forms::Label^ label11;

Guna::UI2::WinForms::Guna2ToggleSwitch^ toggleStyle;

System::Windows::Forms::Label^ label9;

Guna::UI2::WinForms::Guna2ToggleSwitch^ toggleAlwaysHide;

Guna::UI2::WinForms::Guna2Button^ btnSettingsSave;

Guna::UI2::WinForms::Guna2Button^ btnSettingsCancel;

Guna::UI2::WinForms::Guna2Separator^ guna2Separator1;

Bunifu::UI::WinForms::BunifuLabel^ labelSettings;

Bunifu::UI::WinForms::BunifuLabel^ bunifuLabel3;

Bunifu::UI::WinForms::BunifuLabel^ bunifuLabel2;

System::Windows::Forms::Label^ labelBorderBtn;

System::Windows::Forms::Label^ label12;

Guna::UI2::WinForms::Guna2TrackBar^ TrackBorderBtn;

System::Windows::Forms::Label^ labelVolume;

System::Windows::Forms::Label^ label10;

Guna::UI2::WinForms::Guna2TrackBar^ TrackVolume;

System::Windows::Forms::Label^ labelBorderForm;

System::Windows::Forms::Label^ label8;

Guna::UI2::WinForms::Guna2ToggleSwitch^ toggleGreeting;

Guna::UI2::WinForms::Guna2CircleButton^ btnMessageShadow;

System::Windows::Forms::Label^ label7;

Guna::UI2::WinForms::Guna2ToggleSwitch^ toggleShadows;

System::Windows::Forms::Label^ label6;

System::Windows::Forms::Label^ label5;

Guna::UI2::WinForms::Guna2ToggleSwitch^ toggleTransparent;

Guna::UI2::WinForms::Guna2TrackBar^ TrackBorderForm;

System::Windows::Forms::TabPage^ pageProfileEdit;

Guna::UI2::WinForms::Guna2Button^ btnChangePassword;

Guna::UI2::WinForms::Guna2TextBox^ textBoxPassNew2;

Guna::UI2::WinForms::Guna2TextBox^ textBoxPassNew;

Guna::UI2::WinForms::Guna2TextBox^ textBoxPassOld;

System::Windows::Forms::Label^ label3;

Guna::UI2::WinForms::Guna2TextBox^ textBoxCodeNew;

Guna::UI2::WinForms::Guna2Button^ btnCancelChanges;

System::Windows::Forms::LinkLabel^ linkREMOVEACC;

System::Windows::Forms::Label^ lblSexEdit;

System::Windows::Forms::Label^ lblLoginEdit;

System::Windows::Forms::Label^ lblBirthEdit;

System::Windows::Forms::Label^ label4;

System::Windows::Forms::Label^ labelTimer;

Guna::UI2::WinForms::Guna2CirclePictureBox^ pictureBoxCheckCode;

Guna::UI2::WinForms::Guna2CirclePictureBox^ pictureBoxCheckMail;

Guna::UI2::WinForms::Guna2TextBox^ textBoxEmail;

Guna::UI2::WinForms::Guna2TextBox^ textBoxCode;

Guna::UI2::WinForms::Guna2Button^ buttonSendMail;

Guna::UI2::WinForms::Guna2Button^ buttonCheckCode;

System::Windows::Forms::Label^ lblSurnameEdit;

Guna::UI2::WinForms::Guna2TextBox^ textBoxUserSurname;

Guna::UI2::WinForms::Guna2TextBox^ textBoxUserName;

System::Windows::Forms::Label^ lblNameEdit;

Guna::UI2::WinForms::Guna2Button^ btnProfileSave;

Guna::UI2::WinForms::Guna2Button^ guna2Button2;

System::Windows::Forms::PictureBox^ pictureProfileEdit;

System::Windows::Forms::LinkLabel^ linkReMail;

System::Windows::Forms::Label^ label2;

System::Windows::Forms::TabPage^ anim1;

System::Windows::Forms::Label^ lblAnim1;

System::Windows::Forms::TabPage^ anim2;

System::Windows::Forms::Label^ lblAnim2;

System::Windows::Forms::TabPage^ anim4;

Guna::UI2::WinForms::Guna2Button^ btnSync1;

System::Windows::Forms::Label^ lblAnim4;

System::Windows::Forms::TabPage^ anim3;

Guna::UI2::WinForms::Guna2Button^ btnSync;

System::Windows::Forms::Label^ lblSync5;

System::Windows::Forms::Label^ lblSync2;

System::Windows::Forms::Label^ lblSync3;

System::Windows::Forms::Label^ lblSync4;

System::Windows::Forms::Label^ lblSync1;

System::Windows::Forms::Label^ lblSync;

Guna::UI2::WinForms::Guna2Transition^ gunaTransition;

System::Windows::Forms::TabPage^ pageHome;

Bunifu::UI::WinForms::BunifuPages^ PagesTasks;

System::Windows::Forms::TabPage^ TasksMain;

System::Windows::Forms::TabPage^ TaskPage;

Guna::UI2::WinForms::Guna2DragControl^ dragTask;

System::Windows::Forms::RichTextBox^ richTask;

Guna::UI2::WinForms::Guna2Panel^ panelTask;

Guna::UI2::WinForms::Guna2Button^ btnSaveCode;

System::Windows::Forms::Panel^ solutionPanel;

Guna::UI2::WinForms::Guna2Button^ btnTestCode;

System::Windows::Forms::Panel^ panelB;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskAdd;

System::Windows::Forms::Label^ lblLVLB;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskSubtract;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskDivide;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskMultiply;

Guna::UI2::WinForms::Guna2Button^ btnPerviousTask;

Guna::UI2::WinForms::Guna2Button^ btnNextTask;

Guna::UI2::WinForms::Guna2Button^ btnBack;

System::Windows::Forms::Panel^ panelA;

System::Windows::Forms::Label^ label1;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskEvenOrOdd;

System::Windows::Forms::Panel^ panelS;

System::Windows::Forms::Label^ labelA;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskBetterThanAverage;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskReverseSeq;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskPositiveSum;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskHighAndLow;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskSquareDigits;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskGetCount;

System::Windows::Forms::Label^ lblPanelTaskInfo;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskGetMiddle;

Bunifu::Framework::UI::BunifuTileButton^ btnSymmetricPoint;

Bunifu::Framework::UI::BunifuTileButton^ btnTaskGetChar;

System::Windows::Forms::Panel^ panel1;

System::Windows::Forms::Label^ label13;

Guna::UI2::WinForms::Guna2CirclePictureBox^ btnMaximize;

System::Windows::Forms::TextBox^ TaskText;

UI2::WinForms::Guna2MessageDialog^ MessageInfo;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageWarning;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageError;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageQuestion;

System::Windows::Forms::OpenFileDialog^ openFileDialog1;

System::Windows::Forms::Timer^ timerReMail;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageDefault;

Guna::UI2::WinForms::Guna2CirclePictureBox^ pictureUserBar;

System::Windows::Forms::Panel^ panelMain;

System::Windows::Forms::Panel^ panelMenu;

Guna::UI2::WinForms::Guna2BorderlessForm^ borderlessForm;

Guna::UI2::WinForms::Guna2Separator^ separatorMain;

Guna::UI2::WinForms::Guna2VSeparator^ separatorMenu;

Guna::UI2::WinForms::Guna2Button^ btnHome;

Guna::UI2::WinForms::Guna2Button^ btnBook;

Guna::UI2::WinForms::Guna2Button^ btnCourses;

Guna::UI2::WinForms::Guna2Button^ btnMenu;

System::Windows::Forms::Timer^ timerMenu;

Guna::UI2::WinForms::Guna2Button^ btnStats;

Guna::UI2::WinForms::Guna2Button^ btnLogOut;

Guna::UI2::WinForms::Guna2Button^ btnSettings;

Guna::UI2::WinForms::Guna2Button^ btnProfile;

Guna::UI2::WinForms::Guna2PictureBox^ pctrboxLogo;

System::Windows::Forms::Label^ labelPyWave;

Guna::UI2::WinForms::Guna2CirclePictureBox^ btnExit;

Guna::UI2::WinForms::Guna2CirclePictureBox^ btnMinimize;

Guna::UI2::WinForms::Guna2DragControl^ dragMain;

System::Windows::Forms::Panel^ panelUserBar;

System::Windows::Forms::Label^ labelRankBar;

System::Windows::Forms::Label^ labelNameBar;

Guna::UI2::WinForms::Guna2RatingStar^ StarsUser;

System::ComponentModel::IContainer^ components;

System::Windows::Forms::Label^ lblHello;

Guna::UI2::WinForms::Guna2CircleProgressBar^ circleProgress;

System::Windows::Forms::Label^ lblPnlHome3;

System::Windows::Forms::Label^ lblPnlHome2;

System::Windows::Forms::Label^ lblSovet;

Guna::UI2::WinForms::Guna2Elipse^ guna2Elipse1;

Guna::UI2::WinForms::Guna2Elipse^ guna2Elipse2;

Guna::UI2::WinForms::Guna2Elipse^ guna2Elipse3;

Guna::UI2::WinForms::Guna2Elipse^ guna2Elipse4;

Guna::UI2::WinForms::Guna2Elipse^ guna2Elipse5;

Guna::UI2::WinForms::Guna2ShadowPanel^ pnlHome2;

Guna::UI2::WinForms::Guna2ShadowPanel^ pnlHome1;

Guna::UI2::WinForms::Guna2ShadowPanel^ pnlHome3;

Guna::UI2::WinForms::Guna2ShadowPanel^ pnlHome5;

System::Windows::Forms::Label^ lblTimerInApp;

System::Windows::Forms::Label^ lblPnlHome5;

Guna::UI2::WinForms::Guna2Button^ btnReadBook;

System::Windows::Forms::Label^ lblReadBook;

System::Windows::Forms::Label^ label14;

Guna::UI2::WinForms::Guna2Button^ btnRead;

Guna::UI2::WinForms::Guna2ShadowPanel^ pnlHome1n1;

System::Windows::Forms::Label^ lblLastPage;

Guna::UI2::WinForms::Guna2Elipse^ guna2Elipse6;

Guna::UI2::WinForms::Guna2Button^ guna2Button1;

System::Windows::Forms::Timer^ AppTimer;

System::Windows::Forms::Label^ lblStatistic;

Guna::Charts::WinForms::GunaStackedBarDataset^ TasksDataset;

Guna::Charts::WinForms::GunaLineDataset^ StatsDataset;

Guna::UI2::WinForms::Guna2ShadowPanel^ pnlHome4;

Guna::Charts::WinForms::GunaChart^ chartTasksHome;

Bunifu::UI::WinForms::BunifuDropdown^ dropdownReport;

Bunifu::UI::WinForms::BunifuButton::BunifuButton^ btnReport;

System::Windows::Forms::PictureBox^ pictureBox1;

public:

String^ User;

String^ UserRank;

String^ UserSurname;

String^ UserEmail;

String^ UserBirth;

String^ UserSex;

String^ UserName;

array<String^>^ phrases;

ClassFade^ Fade;

public:

MouseEventHandler^ mouseDownHandler;

mainForm(String^ Login)

{

User = Login;

Fade = gcnew ClassFade(this);

InitializeComponent();

mouseDownHandler = gcnew MouseEventHandler(this, &mainForm::Form\_MouseDown);

if (User == "") {

MessageError->Show("Передан некорректный логин", "Ошибка загрузки пользователя");

Application::Exit();

}

this->Load += gcnew EventHandler(this, &mainForm::main\_Load);

srand(static\_cast<unsigned int>(time(NULL)));

}

protected:

~mainForm()

{

if (components)

{

delete components;

}

Application::Exit();

}

#pragma region Windows Form Designer generated code

void InitializeComponent(void){ ... }

#pragma endregion

private:

String^ NEW\_NAME;

String^ NEW\_SURNAME;

String^ NEW\_MAIL;

Boolean isProfileSaved = true;

int securityCode;

int securityCodeNew;

bool email\_confirmed = false;

Boolean isCoursesVisited;

private:

Boolean isDataLoaded;

Void HideWAnimation(Forms::Control^ CONTROL);

Void ShowWAnimation(Forms::Control^ CONTROL);

void logsLoad();

Void labelRankBar\_TextChanged(System::Object^ sender, System::EventArgs^ e);

String^ readBinaryFile(String^ filePath);

size\_t readPassword(String^ filePath);

Void writeBinaryFile(String^ filePath, String^ content);

bool IsValidEmail(String^ email);

int generateSecurityCode();

Void DeleteDirectory(String^ folderPath);

Void DataLoad();

Void DataChange();

Void DataSave();

Void cfgLoad();

Void cfgSave();

String^ SetUserLvl();

private:

Void main\_Load(System::Object^ sender, System::EventArgs^ e);

Void Form\_MouseDown(Object^ sender, MouseEventArgs^ e);

Void btnMenu\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnExit\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnMinimize\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnMaximize\_Click(System::Object^ sender, System::EventArgs^ e);

Void ratingUser\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnQuestionBook\_Click(System::Object^ sender, System::EventArgs^ e);

int secondsSpent;

bool hacks = false; // Экспериментальная переменная для тестовой кнопки

void ReadTimeFromFile();

void WriteTimeToFile();

Void AppTimer\_Tick(System::Object^ sender, System::EventArgs^ e);

private:

bool courseAnimationState = false;

Void SetProfileDefaults();

Boolean CheckSave();

Void timerMenu\_Tick(System::Object^ sender, System::EventArgs^ e);

Void funcSelectTab(TabPage^ tabPage);

Void timerAnim\_Tick(System::Object^ sender, System::EventArgs^ e);

Void btnHome\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnBook\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnCourses\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnProfile\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnStats\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnSettings\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnLogOut\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnSync\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnSync1\_Click(System::Object^ sender, System::EventArgs^ e);

private:

String^ GetFolderCreationDate(String^ folderPath);

Boolean CheckProfileSave(UI2::WinForms::Guna2TextBox^ TB, String^ field);

Void buttonUploadImage\_Click(System::Object^ sender, System::EventArgs^ e);

Void buttonResume\_Click(System::Object^ sender, System::EventArgs^ e);

Void buttonSendMail\_Click(System::Object^ sender, System::EventArgs^ e);

Void buttonCheckCode\_Click(System::Object^ sender, System::EventArgs^ e);

Void timerEmail\_Tick(System::Object^ sender, System::EventArgs^ e);

Void linkReMail\_LinkClicked(System::Object^ sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e);

Void enableMail(bool enable);

Void linkREMOVEACC\_LinkClicked(System::Object^ sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e);

Void btnChangePassword\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnProfileSave\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnCancelChanges\_Click(System::Object^ sender, System::EventArgs^ e);

Void textBoxUserName\_TextChanged(System::Object^ sender, System::EventArgs^ e);

Void textBoxUserSurname\_TextChanged(System::Object^ sender, System::EventArgs^ e);

Void textBoxEmail\_TextChanged(System::Object^ sender, System::EventArgs^ e);

private:

bool menu = true;

bool alwaysHideMenu = true;

bool transparentWhileDrag = true;

bool greeting = true;

bool formShadow = true;

bool PythonWaveStyle = true;

int volume = 50;

int borderBtn = 20;

int borderForm = 25;

private:

void RegisterMouseDownEvent(Control^ parent, bool enable);

Void TrackBorderForm\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e);

Void TrackBorderBtn\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e);

Void TrackVolume\_Scroll(System::Object^ sender, System::Windows::Forms::ScrollEventArgs^ e);

Void toggleTransparent\_CheckedChanged(System::Object^ sender, System::EventArgs^ e);

Void toggleGreeting\_CheckedChanged(System::Object^ sender, System::EventArgs^ e);

Void toggleShadows\_CheckedChanged(System::Object^ sender, System::EventArgs^ e);

Void toggleAlwaysHide\_CheckedChanged(System::Object^ sender, System::EventArgs^ e);

Void btnMessageShadow\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnSettingsSave\_Click(System::Object^ sender, System::EventArgs^ e);

Void toggleStyle\_CheckedChanged(System::Object^ sender, System::EventArgs^ e);

Void PythonWaveStyleState(bool enable);

Void textBoxUserName\_Click(System::Object^ sender, System::EventArgs^ e);

Void textBoxUserSurname\_Click(System::Object^ sender, System::EventArgs^ e);

Void textBoxEmail\_Click(System::Object^ sender, System::EventArgs^ e);

private:

TabPage^ CurrentBookPage;

int CurrentBookIndex;

RichTextBox^ CurrentRichBox;

String^ CurrentBookFile;

void SaveBook(String^ Point, RichTextBox^ richTextBox);

void LoadPage(String^ Point, RichTextBox^ richTextBox);

void LoadBook();

void UpdateLastBookPage();

bool isCustomBookExists();

bool isLastPageExists();

void CreateCustomBook();

void CreateLastPage();

void ChangeSelectionBackColor(Color color, RichTextBox^ richTB);

void ChangeSelectionFontStyle(FontStyle style, RichTextBox^ richTB);

void SetSelectionFontByDefault(RichTextBox^ richTB);

Void btnHighlight\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnUnderline\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnThick\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnStrikeOut\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnItalic\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnClearFilters\_Click(System::Object^ sender, System::EventArgs^ e);

Void dropdownPages\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e);

Void btnPreviousBook\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnBookNext\_Click(System::Object^ sender, System::EventArgs^ e);

private:

int UserProgress;

String^ CurrentTask;

String^ CurrentDifficulty;

void PyRun(String^ code);

Void btnTestCode\_Click(System::Object^ sender, System::EventArgs^ e);

Void TaskText\_TextChanged\_1(System::Object^ sender, System::EventArgs^ e);

void SyntaxHighlight(RichTextBox^ richTB); // Подсветка синтаксиса

Void richTask1\_TextChanged(System::Object^ sender, System::EventArgs^ e);

Void btnSaveCode\_Click(System::Object^ sender, System::EventArgs^ e);

array<String^>^ easyTasks;

array<String^>^ middleTasks;

array<String^>^ hardTasks;

array<String^>^ veryHardTasks;

array<String^>^ GetTaskArray();

int TaskIndex = 0;

Void LoadTask();

void InitializeTaskArrays();

Void NextTask();

Void PreviousTask();

Void btnPerviousTask\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnNextTask\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnBack\_Click(System::Object^ sender, System::EventArgs^ e);

Void lblPanelTaskInfo\_Click(System::Object^ sender, System::EventArgs^ e);

Void panelTask\_Click(System::Object^ sender, System::EventArgs^ e);

void SetTaskAndLoad(String^ task, String^ difficulty);

Void btnTaskAdd\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnTaskDivide\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnTaskMultiply\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnTaskSubtract\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnTaskEvenOrOdd\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnTaskPositiveSum\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnTaskBetterThanAverage\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnReverseSeq\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnTaskGetCount\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnTaskHighAndLow\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnTaskSquareDigits\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnTaskGetChar\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnHero\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnSymmetricPoint\_Click(System::Object^ sender, System::EventArgs^ e);

int GetTasksCompletedCount(int substract);

Void btnReport\_Click(System::Object^ sender, System::EventArgs^ e);

Void SaveReportToFile(String^ reportContent);

private:

String^ currentDayOfWeek;

String^ GetGreetingBasedOnTime();

Void LoadHomePage();

void InitializePhrazes();

String^ GetPhraseForToday();

void InitializePanelBook();

Void btnReadBook\_Click(System::Object^ sender, System::EventArgs^ e);

Void btnRead\_Click(System::Object^ sender, System::EventArgs^ e);

Void guna2Button1\_Click(System::Object^ sender, System::EventArgs^ e); // Тестовая кнопка (LvlUp)

Void mainForm\_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e);

Void mainForm\_Resize(System::Object^ sender, System::EventArgs^ e);

Void buttonQuestion\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ info = "PythonWave\_Справочник.pdf";

if (!String::IsNullOrEmpty(info)) {

try {

Process::Start(info);

}

catch (Win32Exception^ ex) {

MessageError->Show("Ошибка при открытии файла: " + ex->Message, "Ошибка");

}

}

else {

MessageError->Show("Справочная система не найдена", "Ошибка");

}

}

Void Pages\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

if (Pages->SelectedTab->Text == "pageProfileEdit") {

AppTimer->Stop();

}

else {

AppTimer->Start();

}

}

};

}

«ClassTasks.h»

#pragma once

#include <Windows.h>

#include <string>

using namespace System;

using namespace System::IO;

using namespace System::Collections::Generic;

using namespace System::Windows::Forms;

using namespace System::Text;

ref class ClassTasks

{

private:

String^ pathToTasksState;

String^ pathToCounters;

public:

Dictionary<String^, bool>^ taskDict;

Dictionary<String^, DateTime>^ taskCompletionDates;

int completedA\_count = 0;

int completedB\_count = 0;

int completedS\_count = 0;

int completedSplus\_count = 0;

ClassTasks(String^ User) {

if (!String::IsNullOrEmpty(User)) {

pathToTasksState = User + "//tasks\_state.bin";

pathToCounters = User + "//counters.bin";

}

else {

MessageBox::Show("Code: 404\nUser not found", "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

Application::Exit();

return;

}

InitializeTaskDictionary();

if (File::Exists(pathToTasksState)) {

LoadTaskState();

}

else {

SaveTaskState();

}

if (File::Exists(pathToCounters)) {

LoadCounters();

}

else {

SaveCounters();

}

}

String^ GetTaskStates() {

String^ message;

for each (KeyValuePair<String^, bool> kvp in taskDict) {

message += kvp.Key + ": " + kvp.Value.ToString() + "\n";

}

return message;

}

void SolveTask(String^ TaskName) {

SetTaskValue(TaskName, true);

SetTaskCompletionDate(TaskName);

}

bool GetTaskValue(String^ taskName) {

if (taskDict->ContainsKey(taskName)) {

return taskDict[taskName];

}

return false;

}

void SetTaskCompletionDate(String^ taskName) {

array<String^>^ EzTasks = { "add", "multiply", "divide", "subtract", "even\_or\_odd" };

array<String^>^ MidTasks = { "better\_than\_average", "positive\_sum", "reverse\_seq" };

array<String^>^ HardTasks = { "get\_count", "high\_and\_low", "square\_digits" };

array<String^>^ VeryHardTasks = { "get\_char", "symmetric\_point", "get\_middle" };

if (taskDict->ContainsKey(taskName)) {

taskCompletionDates[taskName] = DateTime::Now;

SaveCounters();

SaveTaskState();

}

}

String^ getMaxDifficulty() {

if (completedSplus\_count > 0)

return "Очень сложная";

else if (completedS\_count > 0)

return "Сложная";

else if (completedA\_count > 0)

return "Средняя";

else if (completedB\_count > 0)

return "Легкая";

else

return "Нет выполненных задач";

}

int getCompletedCount(DateTime start, DateTime end) {

return CompletedTasksCount(start, end);

}

private:

void SetTaskValue(String^ taskName, bool value) {

if (taskDict->ContainsKey(taskName)) {

taskDict[taskName] = value;

SaveTaskState();

}

}

void InitializeTaskDictionary() {

taskDict = gcnew Dictionary<String^, bool>();

taskCompletionDates = gcnew Dictionary<String^, DateTime>();

array<String^>^ tasks = { "add", "multiply", "divide", "subtract", "even\_or\_odd",

"better\_than\_average", "positive\_sum", "reverse\_seq",

"get\_count", "high\_and\_low", "square\_digits",

"get\_char", "symmetric\_point", "get\_middle" };

for each (String ^ task in tasks) {

taskDict[task] = false;

}

LoadTaskState();

}

void LoadTaskState() {

if (!File::Exists(pathToTasksState))

return;

array<Byte>^ key = { 0x10, 0x20, 0x30, 0x40, 0x50 };

try {

FileStream^ fs = gcnew FileStream(pathToTasksState, FileMode::Open, FileAccess::Read);

BinaryReader^ br = gcnew BinaryReader(fs);

array<Byte>^ encryptedData = br->ReadBytes((int)fs->Length);

array<Byte>^ decryptedData = gcnew array<Byte>(encryptedData->Length);

for (int i = 0; i < encryptedData->Length; i++) {

decryptedData[i] = encryptedData[i] ^ key[i % key->Length];

}

int index = 0;

array<String^>^ keys = gcnew array<String^>(taskDict->Keys->Count);

taskDict->Keys->CopyTo(keys, 0);

for each (String ^ taskName in keys) {

if (index + sizeof(bool) <= decryptedData->Length) {

taskDict[taskName] = BitConverter::ToBoolean(decryptedData, index);

index += sizeof(bool);

}

else {

break;

}

}

while (index + sizeof(int) <= decryptedData->Length) {

int taskNameLength = BitConverter::ToInt32(decryptedData, index);

index += sizeof(int);

if (index + taskNameLength <= decryptedData->Length) {

String^ taskName = Encoding::UTF8->GetString(decryptedData, index, taskNameLength);

index += taskNameLength;

if (index + sizeof(Int64) <= decryptedData->Length) {

DateTime completionDate = DateTime::FromBinary(BitConverter::ToInt64(decryptedData, index));

index += sizeof(Int64);

taskCompletionDates[taskName] = completionDate;

}

else {

break;

}

}

else {

break;

}

}

br->Close();

fs->Close();

}

catch (Exception^ ex) {

MessageBox::Show("Error loading task state: " + ex->Message, "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

void SaveTaskState() {

array<Byte>^ key = { 0x10, 0x20, 0x30, 0x40, 0x50 };

try {

List<Byte>^ dataBytes = gcnew List<Byte>();

for each (bool value in taskDict->Values) {

array<Byte>^ boolBytes = BitConverter::GetBytes(value);

for each (Byte b in boolBytes) {

dataBytes->Add(b);

}

}

for each (KeyValuePair<String^, DateTime> kvp in taskCompletionDates) {

array<Byte>^ taskNameBytes = Encoding::UTF8->GetBytes(kvp.Key);

array<Byte>^ taskNameLengthBytes = BitConverter::GetBytes(taskNameBytes->Length);

dataBytes->AddRange(taskNameLengthBytes);

dataBytes->AddRange(taskNameBytes);

array<Byte>^ dateBytes = BitConverter::GetBytes(kvp.Value.ToBinary());

dataBytes->AddRange(dateBytes);

}

array<Byte>^ encryptedData = gcnew array<Byte>(dataBytes->Count);

for (int i = 0; i < dataBytes->Count; i++) {

encryptedData[i] = dataBytes[i] ^ key[i % key->Length];

}

FileStream^ fs = gcnew FileStream(pathToTasksState, FileMode::Create, FileAccess::Write);

BinaryWriter^ bw = gcnew BinaryWriter(fs);

bw->Write(encryptedData);

bw->Close();

fs->Close();

}

catch (Exception^ ex) {

MessageBox::Show("Error saving task state: " + ex->Message, "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

void SaveCounters() {

array<Byte>^ key = { 0x10, 0x20, 0x30, 0x40, 0x50 };

try {

List<Byte>^ dataBytes = gcnew List<Byte>();

array<Byte>^ bCountBytes = BitConverter::GetBytes(completedB\_count);

array<Byte>^ aCountBytes = BitConverter::GetBytes(completedA\_count);

array<Byte>^ sCountBytes = BitConverter::GetBytes(completedS\_count);

array<Byte>^ sPlusCountBytes = BitConverter::GetBytes(completedSplus\_count);

dataBytes->AddRange(bCountBytes);

dataBytes->AddRange(aCountBytes);

dataBytes->AddRange(sCountBytes);

dataBytes->AddRange(sPlusCountBytes);

array<Byte>^ encryptedData = gcnew array<Byte>(dataBytes->Count);

for (int i = 0; i < dataBytes->Count; i++) {

encryptedData[i] = dataBytes[i] ^ key[i % key->Length];

}

FileStream^ fs = gcnew FileStream(pathToCounters, FileMode::Create, FileAccess::Write);

BinaryWriter^ bw = gcnew BinaryWriter(fs);

bw->Write(encryptedData);

bw->Close();

fs->Close();

}

catch (Exception^ ex) {

MessageBox::Show("Error saving counters: " + ex->Message, "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

void LoadCounters() {

array<Byte>^ key = { 0x10, 0x20, 0x30, 0x40, 0x50 };

if (!File::Exists(pathToCounters))

return;

try {

FileStream^ fs = gcnew FileStream(pathToCounters, FileMode::Open, FileAccess::Read);

BinaryReader^ br = gcnew BinaryReader(fs);

array<Byte>^ encryptedData = br->ReadBytes((int)fs->Length);

array<Byte>^ decryptedData = gcnew array<Byte>(encryptedData->Length);

for (int i = 0; i < encryptedData->Length; i++) {

decryptedData[i] = encryptedData[i] ^ key[i % key->Length];

}

int index = 0;

completedB\_count = BitConverter::ToInt32(decryptedData, index);

index += sizeof(int);

completedA\_count = BitConverter::ToInt32(decryptedData, index);

index += sizeof(int);

completedS\_count = BitConverter::ToInt32(decryptedData, index);

index += sizeof(int);

completedSplus\_count = BitConverter::ToInt32(decryptedData, index);

index += sizeof(int);

br->Close();

fs->Close();

}

catch (Exception^ ex) {

MessageBox::Show("Error loading counters: " + ex->Message, "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

int CompletedTasksCount(DateTime start, DateTime end) {

int count = 0;

for each (KeyValuePair<String^, DateTime> kvp in taskCompletionDates) {

if (kvp.Value >= start && kvp.Value < end) {

count++;

}

}

return count;

}

};

«ClassProgress.h»

#pragma once

#include <Windows.h>

#include <string>

using namespace System;

using namespace System::IO;

using namespace System::Collections::Generic;

using namespace System::Windows::Forms;

ref class ClassProgress

{

private:

double currentProgress;

String^ pathToProgress;

static int StepB = 3;

static int StepA = 10;

static int StepS = 10;

static int StepSplus = 20;

public:

Dictionary<int, String^>^ ranks;

ClassProgress(String^ User) {

if (!String::IsNullOrEmpty(User))

pathToProgress = User + "//data1.bin";

else {

MessageBox::Show("Code: 404\nUser not found", "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);

Application::Exit();

return;

}

InitializeRanks();

if (File::Exists(pathToProgress)) {

currentProgress = LoadProgress();

}

else {

currentProgress = 0;

SaveProgress();

}

}

static int getTaskPoint(int difficulty) {

switch (difficulty)

{

case 1:

return StepB;

break;

case 2:

return StepA;

break;

case 3:

return StepS;

break;

case 4:

return StepSplus;

break;

default:

return 0;

}

}

int GetCurrentProgress() {

return static\_cast<int>(currentProgress);

}

void SetProgress(int newProgress) {

currentProgress = newProgress;

SaveProgress();

}

String^ GetUserRank() {

for each (KeyValuePair<int, String^> kvp in ranks) {

if (currentProgress < kvp.Key \* 20) {

return kvp.Value;

}

}

return ranks[ranks->Count - 1];

}

int GetUserRankKey() {

int maxKey = 0;

for each (int key in ranks->Keys) {

if (key > maxKey) {

maxKey = key;

}

}

return maxKey;

}

void SolveTaskA() {

currentProgress += StepA;

SaveProgress();

}

void SolveTaskB() {

currentProgress += StepB;

SaveProgress();

}

void SolveTaskS() {

currentProgress += StepS;

SaveProgress();

}

void SolveTaskSplus() {

currentProgress += StepSplus;

SaveProgress();

}

private:

void InitializeRanks() {

ranks = gcnew Dictionary<int, String^>();

ranks->Add(0, "Новичок");

ranks->Add(1, "Исследователь");

ranks->Add(2, "Инженер");

ranks->Add(3, "Разработчик");

ranks->Add(4, "Мастер");

ranks->Add(5, "Эксперт");

}

double LoadProgress() {

FileStream^ fs = gcnew FileStream(pathToProgress, FileMode::Open, FileAccess::Read);

BinaryReader^ br = gcnew BinaryReader(fs);

double progress = br->ReadDouble();

br->Close();

fs->Close();

return progress;

}

void SaveProgress() {

FileStream^ fs = gcnew FileStream(pathToProgress, FileMode::Create, FileAccess::Write);

BinaryWriter^ bw = gcnew BinaryWriter(fs);

bw->Write(currentProgress);

bw->Close();

fs->Close();

}

};

«ClassMail.h»

#pragma once

#include <Windows.h>

#include <iostream>

using namespace System;

using namespace System::Windows;

using namespace System::Windows::Forms;

using namespace System::Net;

using namespace System::Net::Mail;

using namespace System::Net::Security;

using namespace System::Text;

ref class ClassMail

{

private:

Form^ form;

Guna::UI2::WinForms::Guna2MessageDialog^ MessageError;

public:

ClassMail(Form^ f) : form(f) {

MessageError = (gcnew Guna::UI2::WinForms::Guna2MessageDialog());

MessageError->Buttons = Guna::UI2::WinForms::MessageDialogButtons::OK;

MessageError->Caption = L"Ошибка";

MessageError->Icon = Guna::UI2::WinForms::MessageDialogIcon::Error;

MessageError->Parent = form;

MessageError->Style = Guna::UI2::WinForms::MessageDialogStyle::Dark;

MessageError->Text = nullptr;

}

bool SendEmail(String^ to, String^ subject, String^ body) {

try {

MailMessage^ mail = gcnew MailMessage("pythonwaveorg@mail.ru", to, subject, body);

SmtpClient^ client = gcnew SmtpClient("smtp.mail.ru");

client->Port = 587;

client->Credentials = gcnew NetworkCredential("pythonwaveorg@mail.ru", "UN73qgPCSqmSWg7qpUXT");

client->EnableSsl = true;

client->Send(mail);

return true;

}

catch (Exception^ e) {

MessageError->Show(e->Message);

return false;

}

}

};

«ClassFade.h»

#pragma once

#include <Windows.h>

#include <iostream>

using namespace System;

using namespace System::Windows;

using namespace System::Windows::Forms;

using namespace System::Diagnostics;

ref class ClassFade

{

private:

Form^ form; // Ссылка формы

Timer^ timerFade; // Таймер для управления анимацией

int animation = 1;

const double step = 0.05; // step 0.1

const double step1 = 0.2; // step 0.2

public:

ClassFade(Form^ f) : form(f) {

timerFade = gcnew Timer();

timerFade->Interval = 20;

timerFade->Tick += gcnew EventHandler(this, &ClassFade::Animation);

timerFade->Start();

}

void SetAnimation(std::string animationString) {

if (animationString == "in") {

animation = 1;

}

else if (animationString == "minimize") {

animation = 2;

}

else if (animationString == "close") {

animation = 3;

}

else if (animationString == "hide") {

animation = 4;

}

}

void Animation(Object^ sender, EventArgs^ e) {

switch (animation)

{

case 1:

if (form->Opacity < 1)

form->Opacity += step; // Увеличиваем прозрачность

else {

timerFade->Stop(); // Завершение анимации

}

break;

case 2:

if (form->Opacity > 0)

form->Opacity -= step1; // Уменьшаем прозрачность

else {

timerFade->Stop();

form->WindowState = FormWindowState::Minimized;

form->Opacity = 1;

}

break;

case 3:

if (form->Opacity > 0) {

form->Opacity -= step1;

}

else {

timerFade->Stop();

Application::Exit();

form->Close();

}

break;

case 4:

if (form->Opacity > 0) {

form->Opacity -= step1;

}

else {

timerFade->Stop();

form->Hide();

}

break;

default:

break;

}

}

};

Приложение Б

**(справочное)**

**Результаты работы приложения**

На рисунках Б.1 - Б.10 представлены результаты тестирования приложения.

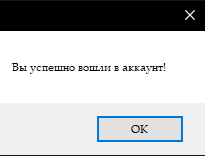


Рисунок Б.1 – Предупреждение об успешном входе

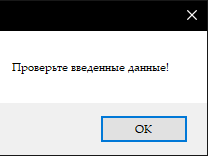


Рисунок Б.2 – Предупреждение об введении некорректных данных

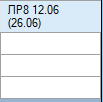


Рисунок Б.3 – Добавление лабораторной работы в таблицу

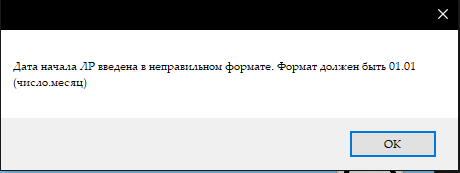


Рисунок Б.4 – Предупреждение об введении некорректных данных

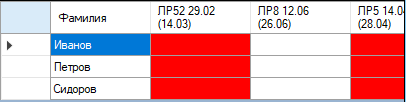


Рисунок Б.5 – Сортировка таблицы

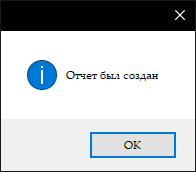


Рисунок Б.6 – Предупреждение об успешном создании отчета

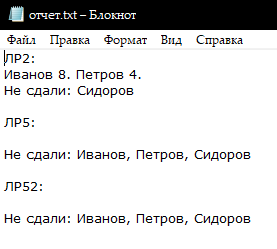


Рисунок Б.7 – Файл отчета

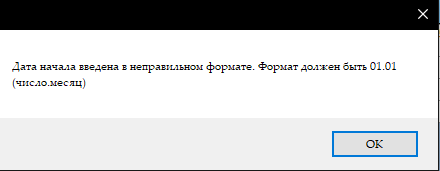


Рисунок Б.8 – Предупреждение об введении некорректных данных

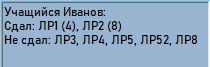


Рисунок Б.9 – Поиск данных из таблицы

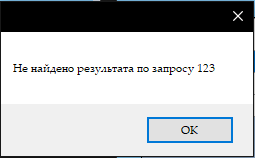


Рисунок Б.10 – Предупреждение об отсутствии результата