**МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ**

**ИМЕНИ И. Т. ТРУБИЛИНА»**

**Факультет прикладной информатики**

**Кафедра системного анализа и обработки информации**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Технологии программирования»

на тему «Разработка программного приложения "Поддержка принятия управленческих решений (разработка системы учёта договоров и расчётов с субподрядчиками), например, материально-технических средств"»

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность «Создание, модификация и сопровождение информационных систем, администрирование баз данных»

Выполнил:

Перваков Артём Иванович

группа ИТ2403

Руководитель:

*К.п.н., доцент кафедры*

*системного анализа и*

*обработки информации*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Самойленкова В. А.

**Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Самойленкова В. А.

**Краснодар 2025**

**Реферат**

!194 с., 27 рис., 10 библ., 23 прил.

Проект, УЧЁТ НАЙМА И ПОДРЯДОВ

**Цель работы:** Разработать приложение для создания и учёта договоров с субподрядчиками с целью увеличения эффективности производственных процессов, а также упрощения поиска и соединения заказчиков и исполнителей.

**Объект исследования:** Процесс взаимодействия заказчиков и подрядчиков.

**Предмет исследования:** Разработка и использование программного приложения для облегчения взаимодействия заказчиков с подрядчиками.

Задачи:

1. Изучить основы работы с C++ и Visual Studio;
2. Проанализировать требования к системе, симулирующей рынок труда;
3. Создать и описать полноценное приложение с реализацией функционала и интерфейса.
4. Продемонстрировать новое приложение и оценить его эффективность.

В данной работе описываются этапы подготовки и разработки приложения «Система учёта договоров», предоставляющего услуги поиска подрядчика и заказчика, просмотра информации о них и учёта заключённых договоров.

# Содержание

[Содержание 3](#_Toc202932970)

[Введение 5](#_Toc202932971)

[Глава 1. Описание предметной области 8](#_Toc202932972)

[1.1 Общие положения 8](#_Toc202932973)

[1.2 Сведения из теории 9](#_Toc202932974)

[1.3 Постановка задачи 11](#_Toc202932975)

[Глава 2. Технология разработки приложения 13](#_Toc202932976)

[2.1 Алгоритм решения 13](#_Toc202932977)

[2.2 Макет приложения 13](#_Toc202932978)

[2.2.1 Форма «Регистрация» 13](#_Toc202932979)

[2.2.2 Форма «Меню» 16](#_Toc202932980)

[2.2.2.1 Меню – Поиск 16](#_Toc202932981)

[2.2.2.2 Меню – Лента 17](#_Toc202932982)

[2.2.2.3 Меню - Аккаунт 18](#_Toc202932983)

[2.2.3. Форма «Информация о пользователе» 19](#_Toc202932984)

[2.3 Описание программы 20](#_Toc202932985)

[2.3.1 Форма Registration 20](#_Toc202932986)

[2.3.2 Форма MainMenu 21](#_Toc202932987)

[2.3.3 Форма UserInfo 22](#_Toc202932988)

[2.4 Результаты работы программы 22](#_Toc202932989)

[Глава 3. Руководство пользователя 27](#_Toc202932990)

[3.1 Подготовка к работе 27](#_Toc202932991)

[3.2 Регистрация 27](#_Toc202932992)

[3.3 Основное меню 27](#_Toc202932993)

[3.3.1. Аккаунт. 27](#_Toc202932994)

[3.3.2. Лента 28](#_Toc202932995)

[3.3.3. Прочее 29](#_Toc202932996)

[3.4 Выход из приложения 30](#_Toc202932997)

[Заключение 31](#_Toc202932998)

[Список использованных источников 33](#_Toc202932999)

[Приложения 34](#_Toc202933000)

# Введение

В современном мире технологии играют ключевую роль в оптимизации и автоматизации бизнес-процессов практически во всех сферах деятельности. Это особенно актуально для рынка труда, где неправильная организация найма напрямую влияет на эффективность работы предприятия. Своевременное получение точной информации о доступных заказах и людях, готовых их выполнить, является основным условием для обеспечения стабильного заработка предприятия, снижения издержек и повышения уровня обслуживания клиентов.

Индустрия найма традиционно ассоциируется с примитивными и часто рутинными операциями, странными ритуалами, долгим просмотром кандидатов, которые при ручном выполнении подвержены ошибкам, неточностям и задержкам. Использование бумажных журналов или простых электронных таблиц (например, в Microsoft Excel) перестает служить современному бизнесу при росте объемов информации. В этих условиях необходимость наличия специализированного программного обеспечения становится очевидной.

Программные приложения для цифровизации найма позволяют ускорить обработку информации, обеспечить прозрачность операций, повысить точность данных и сократить время, затрачиваемое на найм рабочих, ведение отчётов. Более того, программа может подсказать самому предприятию, интересны ли кому-то его товары или услуги.

Актуальность темы данной текущей работы обуславливает широкое распространение наймовой деятельности, а также необходимость выхода малого и среднего бизнеса на рынок ПО. Кроме того, рост электронной коммерции и онлайн-торговли требует быстрой адаптации процесса найма к новым требованиям, таких как найм сотрудников на время, гибкий график работы, выполнение заказов «на лету».

Целью данной работы является разработка программного приложения для учёта договоров с подрядчиками. В рамках достижения поставленной цели предполагается реализовать базовую функциональность, включающую регистрацию и возможность изменения пользовательского аккаунта, отображение подрядов (люди, желающие трудиться за Вашу плату) и заказов (люди, кому интересны плоды вашего труда).

Для достижения целей в курсовой работе решаются следующие задачи:

1. Определить основные и технические характеристики будущего программного продукта;
2. Разработка архитектуры и пользовательского интерфейса приложений;
3. Реализовать программное обеспечение с использованием средств современных разработок;
4. Протестировать систему и проанализировать результаты работы приложения.

**Объектом исследования** является процесс взаимодействия заказчиков и подрядчиков. **Предметом исследования** — разработка и использование программного приложения для облегчения взаимодействия заказчиков с подрядчиками.

**Структура данной работы** включает в себя, кроме *реферата* и *содержания*, *введение* с общими сведениями, связанными с темой; *главу 1* «Теоретические основы автоматизации наймового процесса», в которой определены основные положения об используемых языке и среде программирования, о разрабатываемом приложении, его особенностях и задачах; *главу 2* «Проектирование программного приложения» с описанием решения поставленных задач от общего алгоритма до непосредственного разбора программы и результатов; *главу 3* «Реализация и тестирование программного продукта» и *заключение*, где формулируются выводы по работе. Также в курсовую работу входят *список использованных источников* и *приложение* с листингом программы.

Таким образом, разработка программного приложения для цифровизации найма является актуальной, способствующей оптимизации бизнес-процессов, снижению затрат и повышению качества работы предприятий.

# Описание предметной области

## **Общие положения**

Программное приложение **«Система учёта договоров»** представляет собой актуальное и востребованное решение для автоматизации процессов кратковременного найма. В условиях активного развития торговли и роста экономики всё больше предприятий стремятся снизить затраты, нанимая рабочего не на постоянный срок, а когда он потребуется. Особо важен здесь современный подход для организаций малого и среднего бизнеса, где внедрение простых и эффективных IT-решений влияет на производительность и управляемость ресурсов.

Создание приспособлений, позволяющих вести учёт «токенного найма», активных подрядов и возможных заказов обеспечивает удобный инструмент для работы наймодателя. Простота интерфейса, логическая структура и доступность функционала позволяют использовать систему как техническому персоналу, так и руководителям, заинтересованным в максимизации прибылей.

Курсовой проект включает в себя следующие **этапы реализации поставленных задач:**

1. Обоснование актуальности разработки программного обеспечения для складского учета.
2. Определение цели, предмета, целей и задач курсовой работы.
3. Анализ существующих методов учета и программных средств автоматизации.
4. Определение требований к функционалу и интерфейсу приложения.
5. Разработка структуры системы и структуры базы данных.
6. Реализация основных функций приложения с помощью выбранной среды разработки.
7. Проведение тестирования работоспособности системы и анализ результатов.
8. Подготовка пользовательской документации в приложении.
9. Формулировка выводов по результатам выполнения курсового проекта.

## **Сведения из теории**

В качестве основного языка программирования для разработки приложения был выбран C++ [1] — современный, производительный и объектно-ориентированный язык, обладающий богатым синтаксисом и мощными возможностями для создания настольных приложений. В качестве платформы для построения графического интерфейса была использована Windows Forms, которая отлично интегрируется с Visual Studio и обеспечивает широкий набор элементов управления, упрощающих реализацию визуальной части программы.

Windows Forms позволяет удобно реализовать взаимодействие с использованием таких компонентов, как:

1. Button — для выполнения команд и действий;
2. Label — для отображения текстовой информации и подписей к элементам;
3. TextBox — для ввода и отображения текстовых данных;
4. ComboBox — для выбора значений из списка;
5. DataGridView — для работы с табличными данными, в том числе отображение списка товаров, элементов и операций прихода/расхода;
6. PictureBox — для загрузки и отображения изображений товаров (при необходимости);
7. MenuStrip, TabControl, CheckBox, DateTimePicker и другие элементы позволяют сделать интерфейс гибким и удобным.

Как и полагается объектно-ориентированному языку, в центральные элементы C++ входят классы, которые добавляют объектам свойства, методы, конструкторы и поля. Каждый пользователь, будь то физ. или юрлицо, договор в приложении, представленном в виде отдельного класса, отражает его структуру и поведение. Экземпляры классов — объекты — используются для хранения и обработки конкретных данных.

Пример может служить классом User, включающий поля: name, phone\_number, birth\_date, email, website и т.д., а также методами обработки операций, связями с преобразованием данных дня других методов или добавлением информации о пользователе.

Особое внимание уделяется модификаторам доступа, которые обеспечивают защиту данных и корректное разграничение доступа:

1. public — открытый доступ к членам класса из любого места программы;
2. private — доступ только внутри текущего класса;
3. protected — доступ в текущем и дочерних классах.

Использование направления ООП (наследование, инкапсуляция, полиморфизм) структурирует приложение, делает его легко расширяемым и применимым. Такой подход обеспечивает чистоту кода, логичность построения и удобство в сопровождении системы.

Метод в C++ представляет собой логически завершённый фрагмент кода, выполняющий определённые действия. Метод стандартной структуры включает: модификатор доступа, тип возвращаемого значения, имя метода, список параметров в круглых скобках и инструкции блокировки в фигурных скобках.

Если метод должен вернуть результат, используется ключевое слово return, за которым следует возвращаемое значение. В тех случаях, когда метод не возвращает данные, он объявляется типом void.

## **Постановка задачи**

Приложение, разработанное в рамках данной курсовой работы, носит наименование «Agora». Это прямое заимствование из греческого: агора – главный рынок древнегреческих полисов, где происходила торговля. Основное назначение приложения — автоматизация процессов найма, выполнения заказов и контроля за подрядами. Система обеспечивает сохранение необходимых данных на компьютер пользователя.

Для реализации данного программного продукта был выбран язык C++, который, благодаря своей универсальности, лаконичности и объектно-ориентированному подходу широко используется в области разработки прикладных программ. C++ обеспечивает разработчику множество средств для построения надежных и расширяемых приложений, а его интеграция с технологиями и интерфейсами Windows Forms делает его отличным выбором для создания настольных систем учета.

Разработка велась в среде Visual Studio Community 2022 — одного из наиболее мощных и удобных инструментов для программирования на C++. Это обеспечивает широкий набор инструментов для визуального проектирования, разработки, тестирования окружающей среды и работы с устойчивым контролем и базами данных. Свободность, стабильность и поддержка актуальных стандартов программирования обеспечивают Visual Studio 2022 соответствующие решения для научных и профессиональных проектов. Выбор именно этой версии обусловлен стремлением использовать современные средства разработки, обеспечение долгосрочной актуальности созданного программного обеспечения.

Что касается технических требований, к оборудованию и программному обеспечению, необходимым для работы приложения - они соответствуют системным требованиям Visual Studio 2022. Программа может работать на компьютерах с операционной системой не ниже Windows 10 (желательно редакция «Корпоративная» или «Образовательная»), а также на серверных ОС, начиная с Windows Server 2016. К основным требованиям к «железу» относятся:

1. процессор с архитектурой ARM64 или x64;
2. от 4 ГБ оперативной памяти;
3. 500Мб свободного места на диске.

В качестве системы управления данными в проекте используется нативная библиотека fstream, сохраняющая данные в текстовом формате. Рассматривается переход на JSON-формат.

# Технология разработки приложения

## **Алгоритм решения**

В приложении представлены формы, выполняющие в основном информационную функцию и не содержащие содержательную логику. В связи с этим детальный алгоритм анализа, реализованный с использованием WinForms, будет приведен только для ключевых окон, в которые пользователь вводит данные. К ним относятся: «Регистрация», «Лента», «Аккаунт».

Авторизация реализована автоматически, с помощью чтения данных пользовательского сохранения. Если его нет, открывается окно регистрации.

1. При заполнении всех TextBox’ов становится доступна кнопка регистрации.
2. При регистрации все данные сохраняются в .txt-файл.

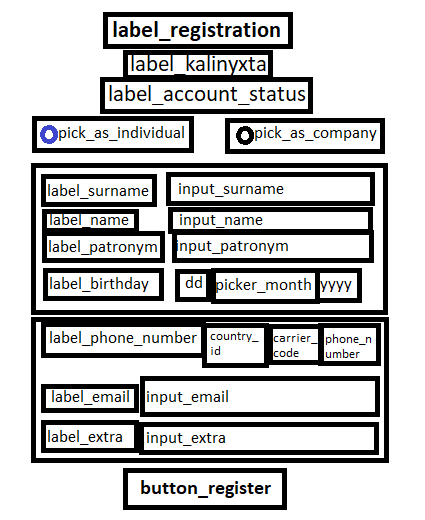
При попадании пользователя на страницу «Аккаунт» появляется необходимость рассмотреть нажатие кнопки «Редактировать»: при её нажатии открывается окно регистрации с уже заполненными полями, которые можно редактировать.

## **Макет приложения**

Как упоминалось ранее в разделе «Алгоритм решения», основные функции реализованы в формах WinForms, где пользователь вводит данные. В связи с этим будет представлен обзор макетов трёх основных страниц приложения, способ оформления которых представлен в лаконичном виде.

#### Форма «Регистрация»

Макет формы «Регистрация» (рисунки 1.1 и 1.2):



*Рисунок 1.1: регистрация физлица*

label\_registration – заголовок.

label\_kalinyxta – «Доброе утро/день/вечер».

label\_account\_status – показывает, найден ли аккаунт в системе.

pick\_as\_individual – выбор регистрации как физлица.

pick\_as\_company – выбор регистрации как юрлица.

label\_surname – «Фамилия:».

input\_surname – ввод фамилии.

label\_name – «Имя:».

input\_name – ввод имени.

label\_patronym – «Отчество:».

input\_patronym – ввод отчества.

label\_birthday – «Дата рождения:».

dd – ввод дня.

picker\_month – выбор месяца.

yyyy – ввод года.

label\_phone\_number – «Номер телефона:».

country\_id - +7, неизменяемый код страны.

carrier\_code\_id – ввод кода оператора.

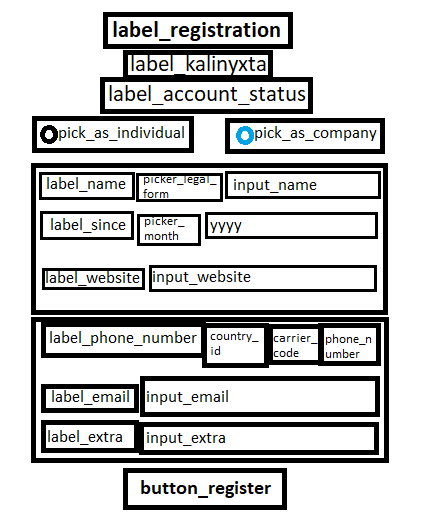
phone\_number – ввод «тела» номера.

label\_email – «email:».

input\_email – ввод электронной почты.

label\_extra – «Дополнительные контакты:».

input\_extra – ввод дополнительных контактов.



*Рисунок 1.2: регистрация юрлица*

label\_registration – заголовок.

label\_kalinyxta – «Доброе утро/день/вечер».

label\_account\_status – показывает, найден ли аккаунт в системе.

pick\_as\_individual – выбор регистрации как физлица.

pick\_as\_company – выбор регистрации как юрлица.

label\_name – «Имя:».

picker\_legal\_form – выбор организационно-правовой формы.

input\_name – ввод имени.

label\_since – «На рынке с:».

picker\_month – выбор месяца.

yyyy – ввод года.

label\_website – «Сайт:»

input\_website – ввод сайта.

label\_phone\_number – «Номер телефона:».

country\_id - +7, неизменяемый код страны.

carrier\_code\_id – ввод кода оператора.

phone\_number – ввод «тела» номера.

label\_email – «email:».

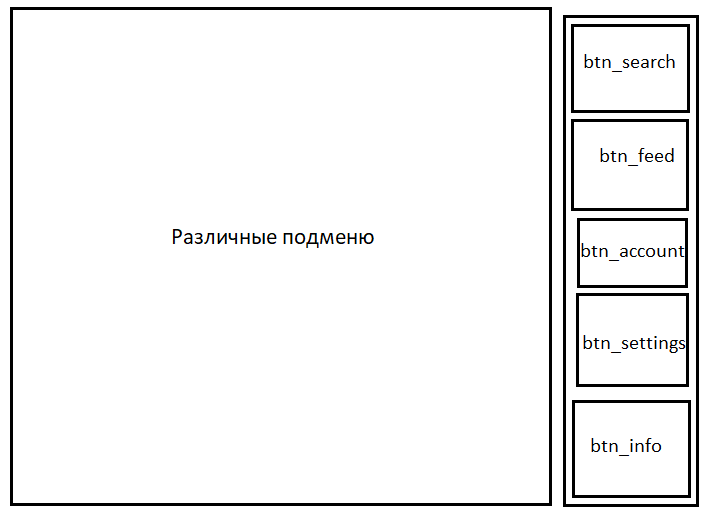
input\_email – ввод электронной почты.

label\_extra – «Дополнительные контакты:».

input\_extra – ввод дополнительных контактов.

#### Форма «Меню»

Макет формы Меню (рисунок 2):



*Рисунок 2*

btn\_search – переход к подменю поиска.

btn\_feed – переход к подменю ленты.

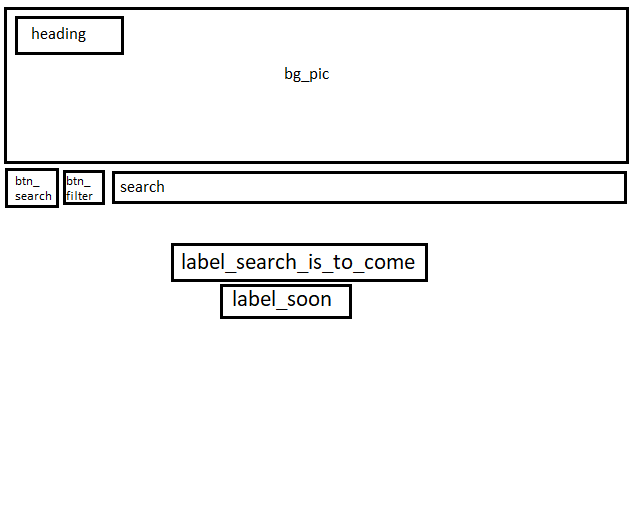
btn\_account – переход к подменю аккаунта.

btn\_settings – переход к подменю настроек; выключен.

btn\_info – MessageBox о приложении.

##### Меню – Поиск

Макет формы «Поиск» (рисунок 3):



*Рисунок 3*

heading – заголовок подменю.

bg\_pic – фоновая картинка.

btn\_search – кнопка поиска; выключена.

btn\_filter – кнопка фильтров; выключена.

search – ввод запроса; выключен.

label\_search\_is\_to\_come – анонс поиска, как функционала.

label\_soon – в будущих обновлениях.

##### Меню – Лента

Макет формы «Лента» (рисунок 4):

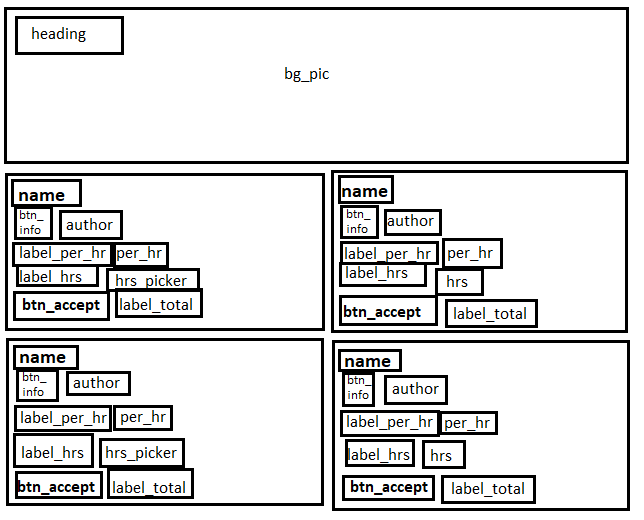


Рисунок 4

heading – заголовок подменю.

bg\_pic – фоновая картинка.

**Для каждого объявления:**

name – название.

btn\_info – открывает форму «Информация об аккаунте» про автора объявления.

author – имя автора объявления.

label\_per\_hr – «За час:».

per\_hr – оплата за час работы.

label\_hrs – «Всего часов:».

hrs – количество часов работы.

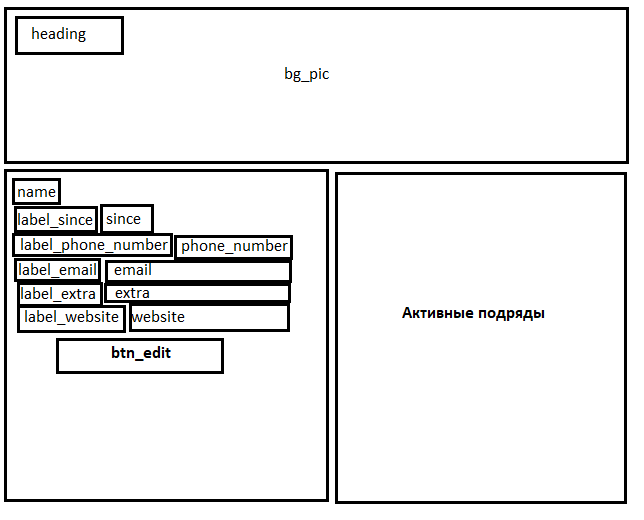
hrs\_picker – выбор количества часов работы (до 8).

btn\_accept – принять заказ/подряд.

label\_total – окончательная оплата; перемножение почасовой на количество часов.

##### Меню - Аккаунт

Макет формы «Аккаунт» (рисунок 5):



*Рисунок 5*

heading – заголовок подменю.

bg\_pic – фоновая картинка.

name – имя пользователя.

label\_since – «Дата рождения:» для физ. лица; «На рынке с:» для юр. лица.

since – дата рождения пользователя или основания компании.

label\_phone\_number – «Номер телефона:».

phone\_number – номер телефона.

label\_email – «email:».

email – эл. почта;

label\_extra – «Доп. контакты:».

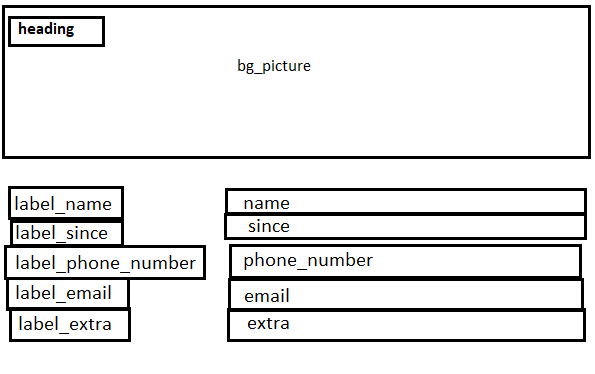
extra – доп. контакты.

label\_website – «Сайт:»

website – сайт.

#### Форма «Информация о пользователе»

Макет формы «Информация о пользователе» (рисунок 6):



*Рисунок 6*

heading – «Информация о пользователе».

bg\_picture – Картинка.

label\_name – «Имя:» / «Название компании:».

name – имя / название компании.

label\_since – «День рождения:» / «На рынке с:».

since – день рождения / основания компании.

label\_phone\_number – «Номер телефона:».

phone\_number – номер телефона.

label\_email – «email:».

email – эл. почта.

label\_extra – «Доп. контакты:».

extra – доп. контакты.

## **Описание программы**

##### Форма Registration

После запуска программы отображается окно «Регистрация» — форма Registration. Интерфейс включает поля для ввода данных, а также кнопку для регистрации.

Программа определит время дня и пожелает пользователю доброго утра/дня/вечера.

У каждого типа аккаунта есть список полей, необходимых к заполнению – без них кнопка регистрации не активируется.

Физлицо: фамилия, имя, дата рождения, телефон.

Юрлицо: название и ОПФ, дата основания, сайт, телефон, e-mail.

Далее, при нажатии кнопки «Регистрация» создаётся объект типа User, который затем сериализуется в user.txt-файл.

##### Форма MainMenu

Форма MainMenu представляет собой приложение общего окна, предоставляющее пользователю доступ к различным разделам системы. В форме конструктора MainForm() представлен InitializeComponent() метод, объединяющий все элементы интерфейса, включая меню, кнопки, панель и другие визуальные компоненты.

Форма MainForm используется переходным хабом и некоей площадкой, на которой пользователь может перейти к подменю системы:

1. Кнопка search – Переход в раздел «Поиск»
2. Кнопка feed – Раздел «Лента».
3. Кнопка settings – «Настройки» (выключенные).
4. Пункт меню "Информация", показывающий инфо о программе в MessageBox.

Во всех перечисленных случаях текущая форма MainForm остаётся внизу, перемещая подменю между собой с помощью метода SendToFront.

По размерам окно фиксировано, т.к. Windows Forms не поддерживают возможность ресайзинга окна; перемещать же окно можно куда угодно.

###### 2.3.2.1. Подменю «Поиск»

Это меню статично, так как не было выделено время на его реализацию. Добавление такого функционала будет означать кардинальный передел структуры программы.

###### 2.3.2.2. Подменю «Лента»

В этом меню 4 случайных подряда, генерируемых ф-ей generate\_random\_listing(), результаты которой синхронизируются с интерфейсом через объекты Listing\_Interface. Результаты генерации всегда можно обновить кнопкой update, обновление данных происходит за 0.5с.

При создании объекта Listing\_Interface передаются все ссылки на нужные элементы интерфейса. Основная задача объекта – связывать внутренние переменные Listing со внешним интерфейсом.

###### 2.3.2.3. Подменю «Аккаунт»

Слева на экран выводится информация о пользователе, а справа – его активные контракты. Если их нет, высвечивается Label, говорящий, что можно посмотреть в Ленте.

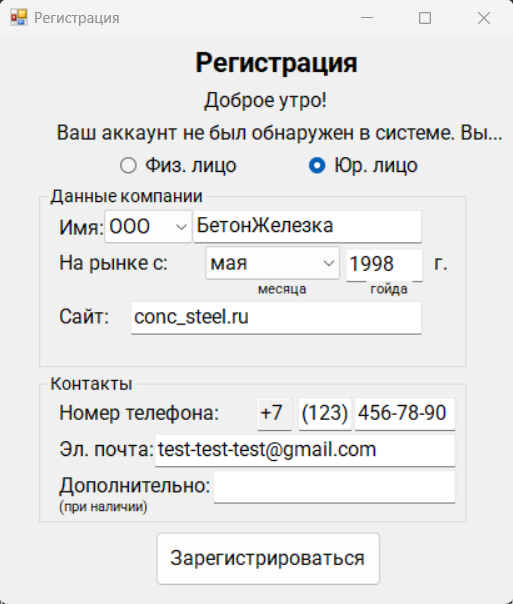
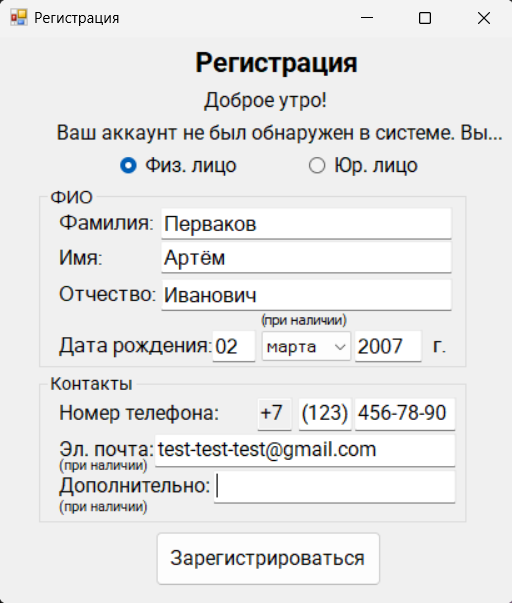
Данные аккаунта можно редактировать прямо здесь. Для этого при нажатии кнопки «Редактировать» загружается форма Registration с уже введёнными данными, которые можно редактировать и сохранить.

##### Форма UserInfo

Статичная форма, выводящая все известные данные о пользователе без права редактировать. Пользователь передаётся через параметр user в конструкторе.

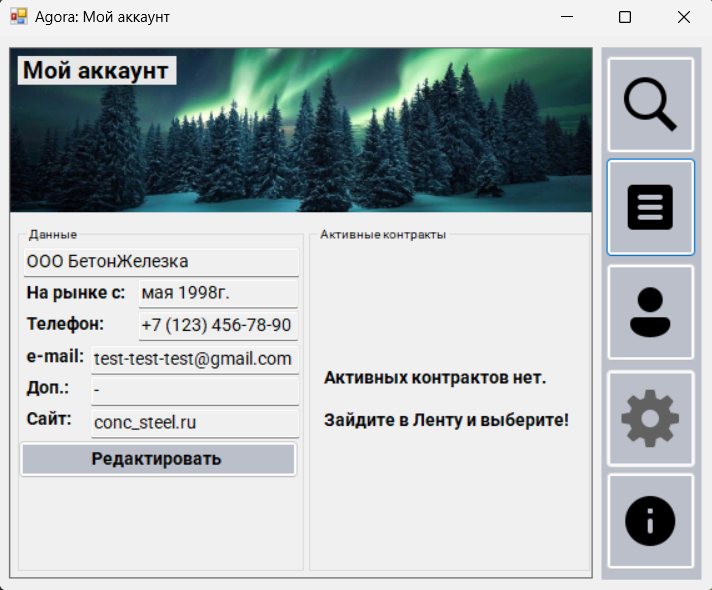
## **Результаты работы программы**

При открытии приложения оно пытается прочитать данные пользователя. Если они не найдены, на экране появляется окно «Регистрация». При вводе данных (рисунок 7, 8) пользователь сможет зарегистрировать свой аккаунт:



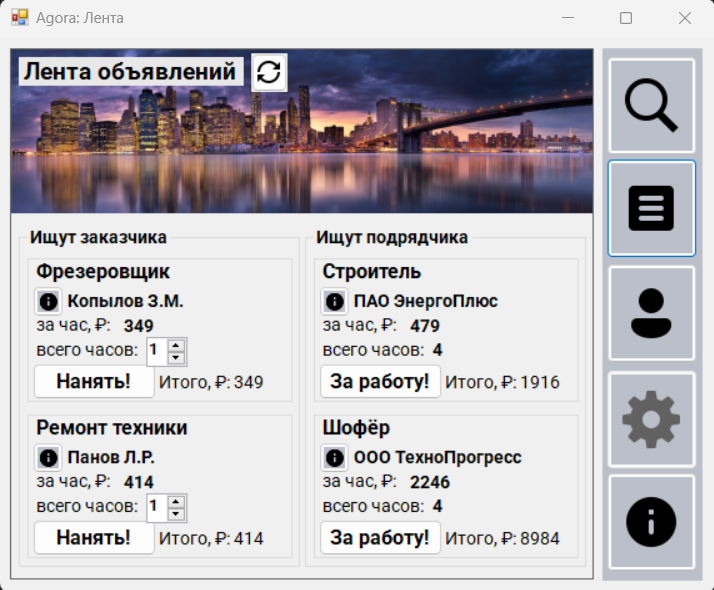
*Рисунки 7, 8*

Если аккаунт был зарегистрирован заранее, или пользователь прошёл процесс регистрации, он попадает в главное меню – подменю аккаунта (рисунок 9):



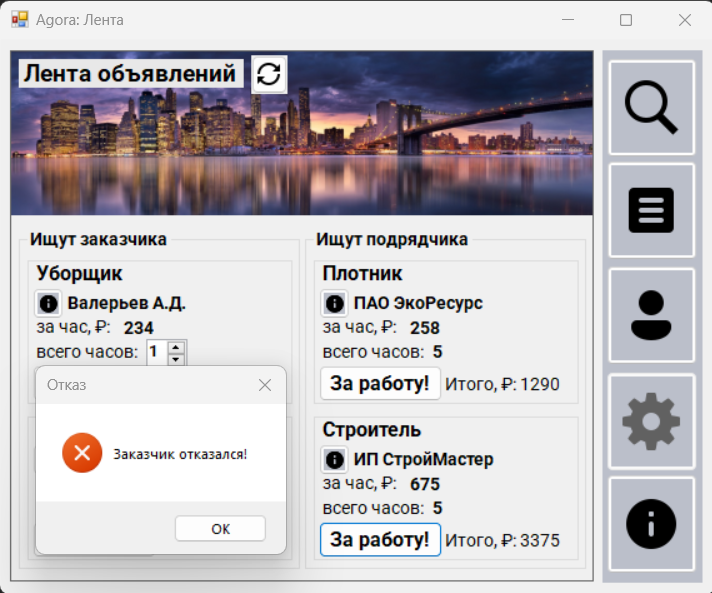
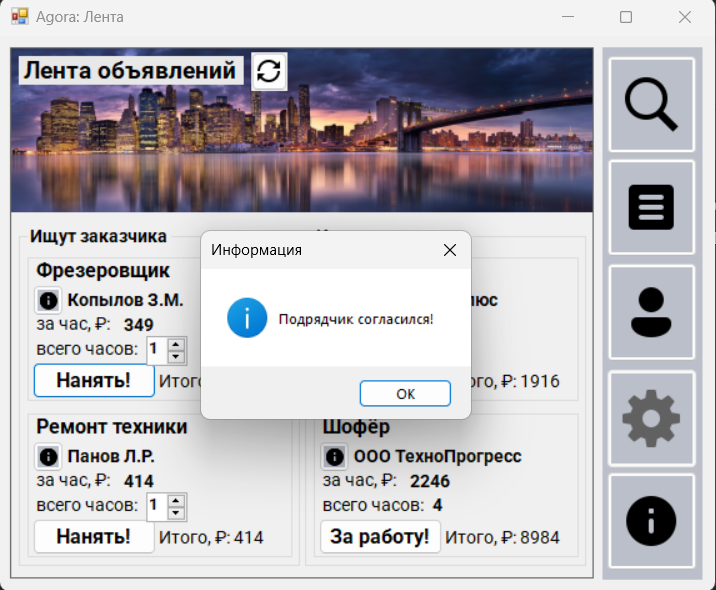
*Рисунок 9*

Если пользователю захочется найти контракт, он должен будет открыть подменю «Лента» (рисунок 10):



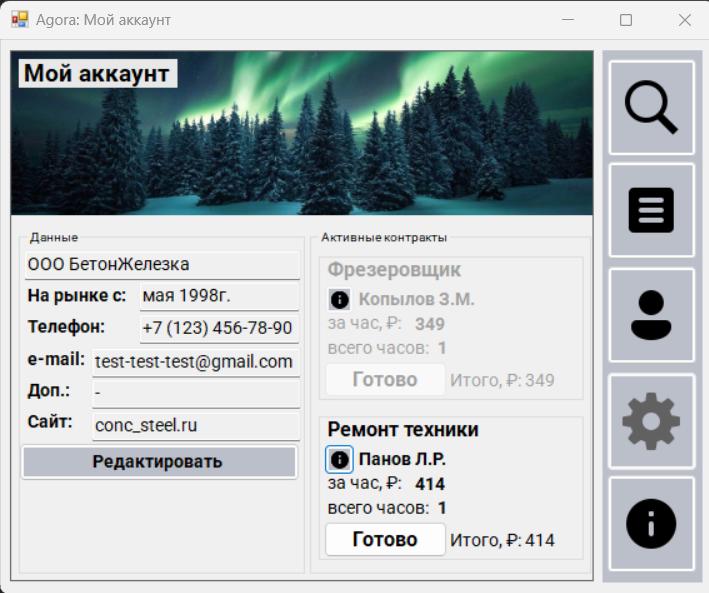
*Рисунок 10*

Тут он сможет выбрать подходящий контракт – причём подрядчик может как согласиться (рисунок 11), так и отказаться (рисунок 12):



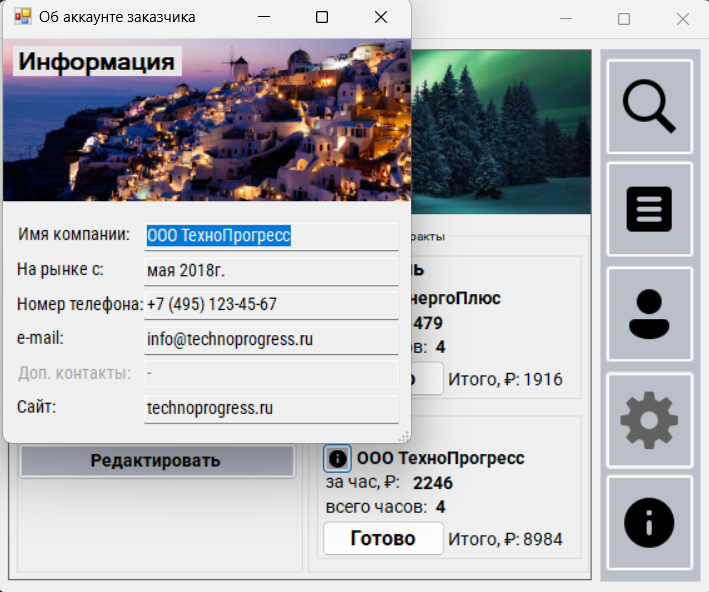
*Рисунки 11, 12*

Заключённые контракты можно посмотреть у себя в аккаунте – и там же отметить выполненными (рисунок 13):



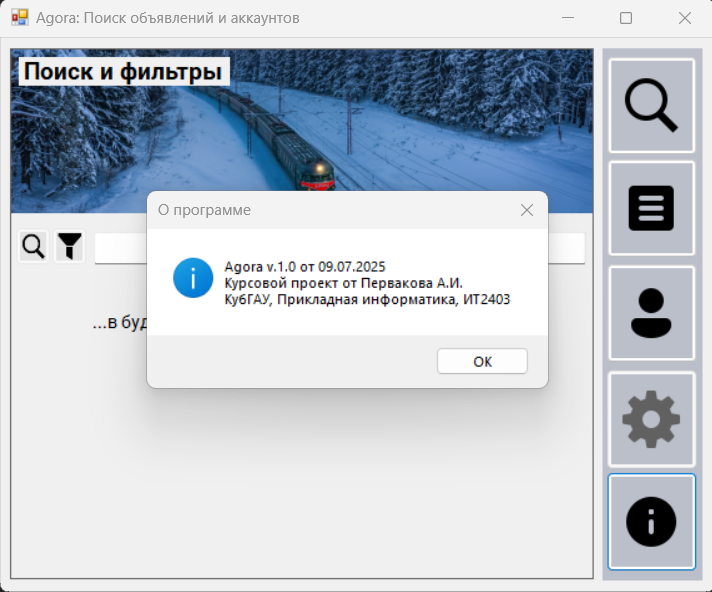
*Рисунок 13*

При нажатии на кнопку info покажется информация о авторе объявления (рисунок 14). То же самое возможно и в Ленте.



*Рисунок 14*

При нажатии же на кнопку info в панели навигации покажется информация о программе (рисунок 15):



*Рисунок 15*

# Руководство пользователя

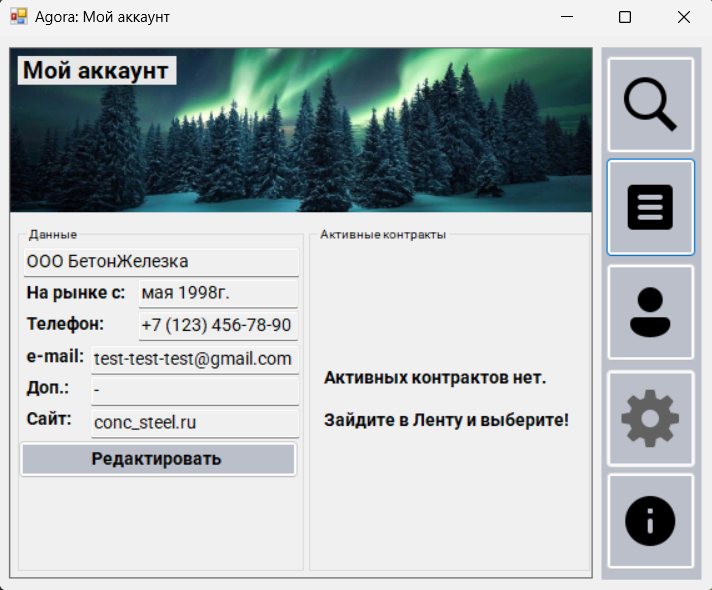
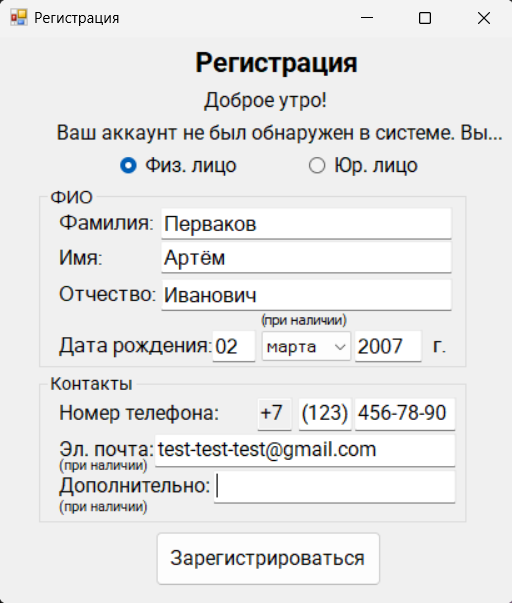
## **Подготовка к работе**

Для работы с программой «Agora» скачайте архив с исполняемым файлом. Для запуска открывайте файл Agora.exe, а лучше сделайте на него ярлык на Рабочем столе.

**Не перемещайте файл .exe вне папки!** В ней содержатся необходимые для работы файлы.

## **Регистрация**

При первом запуске открывается страница «Регистрация» (рисунок 16). Введите свои данные (они хранятся локально – никто не украдёт), после чего попадёте на главную страницу приложения (рисунок 17):



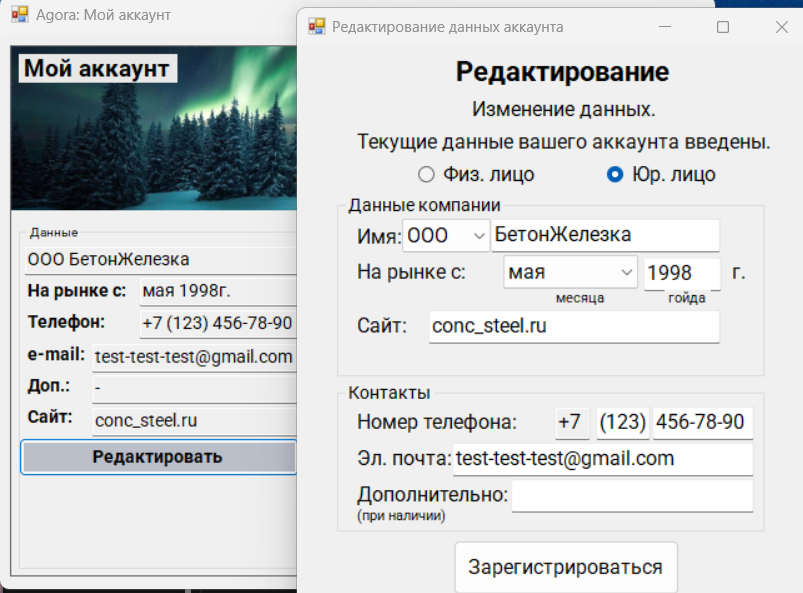
Рисунки 16, 17.

## **Основное меню**

Страница «Основное меню» является основной, связующей между остальными окнами. Справа есть полоса навигации, перемещающая вас в меню «Поиск», «Лента», «Аккаунт», «Информация».

#### 3.3.1. Аккаунт.

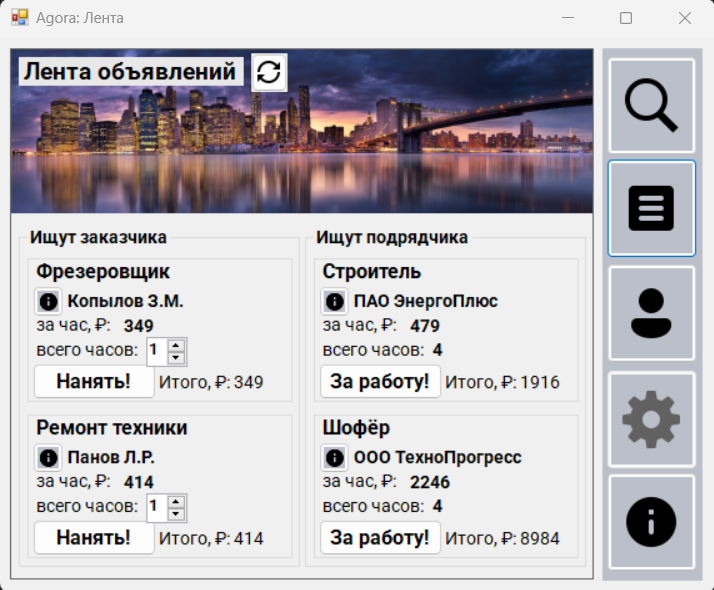
Страница по умолчанию (рисунок 17). Показывает активные подряды, даёт редактировать данные своего аккаунта (рисунок 18):



*Рисунок 18*

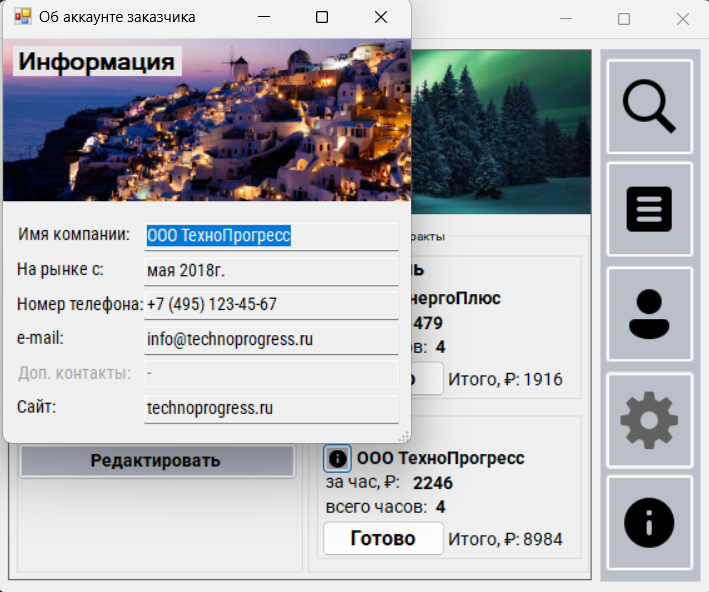
#### 3.3.2. Лента

В «Ленте» вы можете найти новые подряды – просто кликните на тот, что вам нравится. Если таких нет – нажмите кнопку «Обновить» справа от заголовка (рисунок 18):



*Рисунок 18*

Вы можете просмотреть информацию о любом из авторов – просто нажмите кнопку "Инфо" слева от их имени (рисунок 19):



*Рисунок 19*

#### 3.3.3. Прочее

Меню «Поиск» пока что неактивно, а кнопка «Информация» показывает информацию о приложении (рисунок 20):

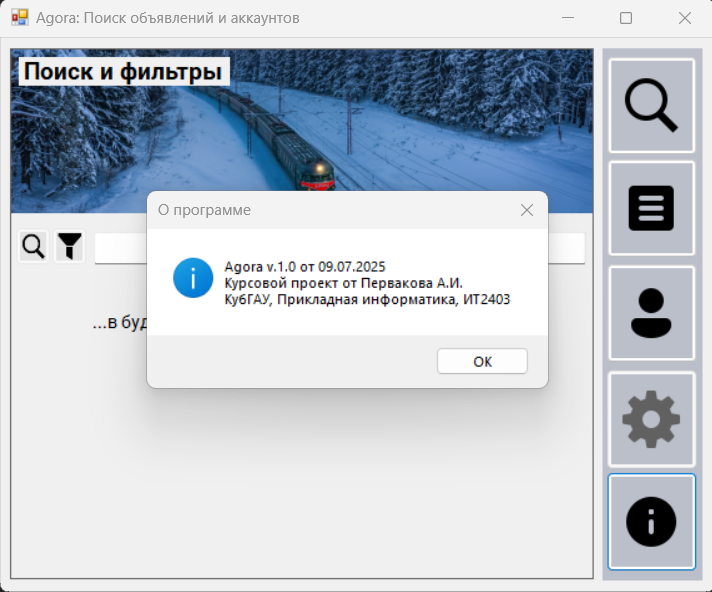


Рисунок 20

## **Выход из приложения**

Закрытие приложения осуществляется нажатием «крестика» в правом верхнем углу на любой странице. При закрытии приложения происходит автоматическое сохранение всех необходимых данных и сброс всех данных, введённых в текстовые поля. При повторном открытии приложения оно будет иметь первоначальный вид, а авторизация произойдёт автоматически.

# Заключение

В ходе выполнения курсового проекта было разработано программное приложение «Agora», основной целью которого является автоматизация процесса найма подрядчиков, слежка за их контрактами. Дополнительно реализованы формы регистрации и информации о пользователях, обеспечение безопасной работы с введёнными данными.

В процессе выполнения работы были решены ключевые задачи, включая проектирование структуры приложения, реализацию графического интерфейса с использованием Windows Forms, и построение логики взаимодействия между пользовательским интерфейсом и хранилищем данных. Программа поддерживает основные операции над записями: добавление новых материалов, изменение существующих данных, удаление сохранённых данных. Все эти функции доступны через интерфейс, понятны даже неподготовленному новому пользователю.

Особое внимание в процессе разработки было уделено стабильности и расширяемости приложения. В коде реализованы принципы модульности, которые предусматривают сопровождение и потенциальную модернизацию системы. Программа готова обработать повреждённые данные и даже показать, где конкретно они неправильны.

Использование технологий C++ и Windows Forms создаёт прочные технологические основы, которые позволяют адаптировать определенное решение под другие сферы применения — например, площадка для купли-продажи.

Созданное программное обеспечение может быть внедрено в предпринимательство и производственные учреждения, а также на малые и средние предприятия. Простота установки, дружественный интерфейс и наличие детального руководства пользователя делают его доступным для широких кругов, включая тех людей, у кого отсутствуют глубокие знания в области информационных технологий.

Таким образом, приложение «Agora» представляет собой функциональное, надёжное и готовое к использованию решение, позволяющее значительно сократить и корректировать работу с данными в найме, подрядах или похожей области. Его потенциал для дальнейшего развития открывает широкие перспективы для внедрения в различные сферы бизнеса.

# Список использованных источников

1. Head First. Изучаем C#. 4-е изд. / Пер. с англ. Е. Матвеева. — СПб. : Питер, 2022. — 768 с.: ил. — (Серия «Head First O’Reilly»).
2. Краткий обзор языка C# // Официальный сайт корпорации Майкрософт [Электронный ресурс]. – Режим доступа : https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/. – Яз. рус., англ.
3. Документация по Visual Studio // Официальный сайт корпорации Майкрософт [Электронный ресурс]. – Режим доступа : https://learn.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2022&preserve-view=true. – Яз. рус., англ.
4. Документация по .NET // Официальный сайт корпорации Майкрософт [Электронный ресурс]. – Режим доступа : https://learn.microsoft.com/ru-ru/dotnet/?WT.mc\_id=dotnet-35129-website#create-your-application. – Яз. рус., англ.
5. Head First. Изучаем C#. 4-е изд. / Пер. с англ. Е. Матвеева. — СПб. : Питер, 2022. — 768 с.: ил. — (Серия «Head First O’Reilly»).
6. Краткий обзор языка C# // Официальный сайт корпорации Майкрософт [Электронный ресурс]. – Режим доступа : https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/. – Яз. рус., англ.
7. Документация по Visual Studio // Официальный сайт корпорации Майкрософт [Электронный ресурс]. – Режим доступа : https://learn.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2022&preserve-view=true. – Яз. рус., англ.
8. Документация по .NET // Официальный сайт корпорации Майкрософт [Электронный ресурс]. – Режим доступа : https://learn.microsoft.com/ru-ru/dotnet/?WT.mc\_id=dotnet-35129-website#create-your-application. – Яз. рус., англ.

# Приложения

#### Листинг кода name.hpp

#pragma once

#ifndef NAME\_HPP

#define NAME\_HPP

#include <string>

#include <vector>

#include "../../utils.hpp"

class Name; /// abstract class

class Individual\_Name;

class Company\_Name;

class Name

{

public:

Name(const std::wstring& \_name)

: name(\_name)

{}

virtual std::wstring as\_filename(const std::wstring& extension = L".txt") = 0;

virtual std::vector<std::wstring> as\_vector() = 0;

virtual std::wstring get\_normal() = 0;

virtual std::wstring get\_full() = 0;

virtual std::wstring get\_short() = 0;

protected:

std::wstring name;

};

class Individual\_Name: public Name

{

public:

Individual\_Name(const std::vector<std::wstring>& separated\_raw\_data)

: surname(separated\_raw\_data[0]), Name(separated\_raw\_data[1]), patronym(separated\_raw\_data[2])

{}

Individual\_Name(const std::wstring& \_name, const std::wstring& \_surname, const std::wstring& \_patronym = L"")

: Name(\_name), surname(\_surname), patronym(\_patronym)

{}

std::vector<std::wstring> as\_vector()

{

const std::vector<std::wstring> result = {surname, name, patronym};

return result;

}

/// ivanov\_ivan\_ivanovich<...>

std::wstring as\_filename(const std::wstring& extension = L".txt")

{

return translit\_CtoL(surname) + L'\_' + translit\_CtoL(name) + L'\_' + translit\_CtoL(patronym) + extension;

}

/// The form that'll appear in interface. Returns 'get\_short()'

std::wstring get\_normal()

{

return get\_short();

}

/// Иванов Иван Иванович

std::wstring get\_full()

{

if (patronym == L"")

return surname + L' ' + name;

return surname + L' ' + name + L' ' + patronym;

}

/// Иванов И.И.

std::wstring get\_short()

{

if (patronym == L"")

return surname + L' ' + name[0] + L'.';

return surname + L' ' + name[0] + L'.' + patronym[0] + L'.';

}

private:

std::wstring surname,

patronym;

friend std::wstring \_get\_email(const Individual\_Name& name);

};

class Company\_Name : public Name

{

public:

Company\_Name(const std::vector<std::wstring>& separated\_raw\_data)

: legal\_form(separated\_raw\_data[0]), Name(separated\_raw\_data[1])

{

// adding the rest of the name. Kinda dirty code

for (int i = 2; i < separated\_raw\_data.size(); i++)

name += L" " + separated\_raw\_data[i];

}

Company\_Name(const std::wstring& \_legal\_form, const std::wstring& \_name)

: legal\_form(\_legal\_form), Name(\_name)

{}

std::vector<std::wstring> as\_vector()

{

const std::vector<std::wstring> result = {legal\_form, name};

return result;

}

/// tmyvbabla<...>

std::wstring as\_filename(const std::wstring& extension = L".txt")

{

return translit\_CtoL(name) + extension;

}

/// The form that'll appear in interface. Returns 'get\_full()'

std::wstring get\_normal()

{

return get\_full();

}

/// ООО "Тмывбабла"

std::wstring get\_full()

{

return legal\_form + L' ' + name; /// it's already wrapped in quote marks

}

/// Тмывбабла

std::wstring get\_short()

{

return name;

}

private:

std::wstring legal\_form;

};

#endif

#### Листинг кода phone\_number.hpp

#pragma once

#ifndef PHONE\_NUMBER\_HPP

#define PHONE\_NUMBER\_HPP

#include <string>

#include <vector>

#include "../../utils.hpp"

/// - The concept of a phone number.

class Phone\_Number;

class Phone\_Number

{

public:

Phone\_Number(const std::vector<std::wstring>& separated\_raw\_data)

: country\_code(separated\_raw\_data[0]), carrier\_code(separated\_raw\_data[1]), body(separated\_raw\_data[2])

{}

Phone\_Number(const std::wstring& \_country\_code, const std::wstring& \_carrier\_code, const std::wstring& \_body)

: country\_code(\_country\_code), carrier\_code(\_carrier\_code), body(\_body)

{}

// +7 (123) 456-78-90

std::wstring as\_text()

{

std::wstring number\_formatted = body;

number\_formatted.insert(3, L"-");

number\_formatted.insert(6, L"-"); // offset of 1 from the previous insert

return country\_code + L" (" + carrier\_code + L") " + number\_formatted;

}

// +7 (123) \*\*\*-\*\*-90

std::wstring conceal()

{

std::wstring concealed\_num = this->as\_text();

for (unsigned int i = concealed\_num.size() - 4; i >= concealed\_num.size() - 9; i--)

{

if (concealed\_num[i] != L'-')

concealed\_num[i] = L'\*';

}

return concealed\_num;

}

// +7 123 4567890

std::wstring serialize()

{

return country\_code + L' ' + carrier\_code + L' ' + body;

}

private:

std::wstring country\_code, carrier\_code, body;

};

#endif

#### Листинг кода listing.hpp

#pragma once

#ifndef LISTING\_HPP

#define LISTING\_HPP

#include <string>

#include <unordered\_map>

#include "user.hpp"

#define CUSTOMER\_LISTING 1

#define CONTRACTOR\_LISTING 2

class Listing

{

public:

std::wstring name;

unsigned int per\_hr;

unsigned int hrs;

/// подрядчик | заказчик | автор

User \*contractor, \*customer, \*author;

Listing& operator=(const Listing& other)

{

if (this != &other)

{

name = other.name;

per\_hr = other.per\_hr;

hrs = other.hrs;

contractor = other.contractor;

customer = other.customer;

author = other.author;

}

return \*this;

}

bool operator==(const Listing& other)

{

return (

name == other.name

&& per\_hr == other.per\_hr

&& hrs == other.hrs

&& contractor == other.contractor

&& customer == other.customer

&& author == other.author

);

}

Listing(const std::wstring& \_name, const unsigned int \_hrs, const unsigned int \_per\_hr,

User\* \_contractor = nullptr, User\* \_customer = nullptr, User\* \_author = nullptr)

: name(\_name), hrs(\_hrs), per\_hr(\_per\_hr),

contractor(\_contractor), customer(\_customer), author(\_author)

{

if (author != nullptr) /// author explicitly set

return;

if (contractor != nullptr)

author = contractor;

else if (customer != nullptr)

author = customer;

else

show\_error(L"Listing with no users."); // can be bypassed by explicitly setting author

}

/// For compiler not to whine.

Listing(){}

unsigned int payment\_total()

{

return hrs \* per\_hr;

}

/// Assumes every contract involves 'user' - inter-NPC contracts impossible.

std::wstring serialize() /// 'user' for comparing and finding where is 'user.txt' needed as a savefile

{

std::wstring result =

name + L'\n'

+ std::to\_wstring(per\_hr) + L'\n'

+ std::to\_wstring(hrs) + L'\n';

if (contractor == nullptr)

result += L'\n';

else if (contractor == author)

result += contractor->name->as\_filename() + L'\n';

else

result += L"user.txt\n";

if (customer == nullptr)

result += L'\n';

else if (customer == author)

result += customer->name->as\_filename() + L'\n';

else

result += L"user.txt\n";

return result;

}

};

#### Листинг кода random\_individual\_generator.hpp

#pragma once

#ifndef RANDOM\_INDIVIDUAL\_GENERATOR\_HPP

#define RANDOM\_INDIVIDUAL\_GENERATOR\_HPP

#include <string>

#include <vector>

#include "user.hpp"

#include "../utils.hpp"

/// Here be:

Individual\* get\_random\_individual();

std::wstring \_get\_email(const Individual\_Name& name);

Phone\_Number \_get\_random\_phone\_number(const bool corporate = false);

Individual\_Name \_get\_random\_individual\_name();

Individual\* get\_random\_individual()

{

Individual\_Name name = \_get\_random\_individual\_name();

Date birth\_date = \_get\_random\_date();

Phone\_Number number = \_get\_random\_phone\_number();

std::wstring email = \_get\_email(name);

Individual\* result = new Individual(name, birth\_date, number, email);

return result;

}

#pragma region = Names =

const std::wstring names[30] = {

L"Александр", L"Артём", L"Вадим", L"Виктор", L"Георгий", L"Глеб", L"Даниил", L"Данил", L"Дмитрий", L"Евгений",

L"Егор", L"Захар", L"Иван", L"Илья", L"Кирилл", L"Константин", L"Лев", L"Леонид", L"Марк", L"Матвей",

L"Мирон", L"Михаил", L"Никита", L"Пётр", L"Роман", L"Руслан", L"Савелий", L"Степан", L"Тихон", L"Фёдор"

};

// PROBLEM: "Тимофеев" is causing problems (???)

const std::wstring surnames[30] = {

L"Андреев", L"Беляев", L"Васильев", L"Валерьев", L"Волков", L"Высоцкий", L"Горохов", L"Гришин", L"Громов", L"Зайцев",

L"Захарченко", L"Иванов", L"Киселев", L"Козлов", L"Копылов", L"Миронов", L"Моисеев", L"Носов", L"Орлов", L"Павлов",

L"Панов", L"Петров", L"Семенов", L"Смирнов", L"Соловьев", L"Степанов", L"Тимофеев", L"Трофимов", L"Устинов", L"Яковлев"

};

const std::wstring patronyms[30] = {

L"Александрович", L"Артёмович", L"Вадимович", L"Викторович", L"Георгиевич", L"Глебович", L"Даниилович", L"Данилович", L"Дмитриевич", L"Евгеньевич",

L"Егорович", L"Захарович", L"Иванович", L"Ильич", L"Кириллович", L"Константинович", L"Леонидович", L"Маркович", L"Матвеевич", L"Макарович"

L"Миронович", L"Михайлович", L"Никитович", L"Пётрович", L"Романович", L"Русланович", L"Савельевич", L"Степанович", L"Тихонович", L"Фёдорович"

};

Individual\_Name \_get\_random\_individual\_name()

{

std::wstring name = names[randint(29)];

std::wstring surname = surnames[randint(29)];

std::wstring patronym = patronyms[randint(29)];

Individual\_Name result(name, surname, patronym);

return result;

}

#pragma endregion

#pragma region = Contacts =

const std::wstring \_email\_domains[3] = {L"gmail.com", L"mail.ru", L"yandex.ru"};

/// Random e-mails are ONLY FOR INDIVIDUALSs and are generated according to the name transliteration.

std::wstring \_get\_email(const Individual\_Name& name)

{

// igor\_ivanov245@domain.com

std::wstring body = translit\_CtoL(name.name) + L'\_'

+ translit\_CtoL(name.surname)

+ std::to\_wstring(randint(100, 999));

std::wstring domain = \_email\_domains[randint(2)];

return body + L'@' + domain;

}

// | MTS | BeeLine | Tele2 | Misc |

const int \_carrier\_code\_limits[8][2] = {

{910, 919}, {903, 909}, {900, 908}, {920, 937},

{980, 989}, {960, 976}, {991, 999}, {800, 800}

};

Phone\_Number \_get\_random\_phone\_number(const bool corporate/\* = false\*/)

{

int operator\_type = randint(6 + corporate); // so that individuals don't get a 800 [i = 7] operator code

int code = randint(\_carrier\_code\_limits[operator\_type][0], \_carrier\_code\_limits[operator\_type][1]);

std::wstring carrier\_code = std::to\_wstring(code);

std::wstring body;

for (int i = 0; i < 7; i++)

body += (L'0' + randint(9));

Phone\_Number result(L"+7", carrier\_code, body);

return result;

}

#pragma endregion

#endif

#endif

#### Листинг кода random\_listing\_generator.hpp

#pragma once

#ifndef RANDOM\_LISTING\_GENERATOR

#define RANDOM\_LISTING\_GENERATOR

#include <string>

#include <vector>

#include "listing.hpp"

#include "user.hpp"

#include "random\_individual\_generator.hpp"

#include "../save\_load.hpp"

#include "../utils.hpp"

/// Here be:

class \_Job\_Class;

Listing get\_random\_listing(int job\_type);

class \_Job\_Class

{

public:

\_Job\_Class(const unsigned int \_per\_hr\_LOW, const unsigned int \_per\_hr\_HIGH, const std::vector<std::wstring>& \_job\_names)

: per\_hr\_lower\_bound(\_per\_hr\_LOW), per\_hr\_upper\_bound(\_per\_hr\_HIGH), job\_names(\_job\_names)

{}

std::wstring random\_job\_name()

{

return job\_names[randint(0, job\_names.size() - 1)];

}

int random\_per\_hr()

{

return randint(per\_hr\_lower\_bound, per\_hr\_upper\_bound);

}

private:

const unsigned int per\_hr\_lower\_bound;

const unsigned int per\_hr\_upper\_bound;

const std::vector<std::wstring> job\_names;

};

\_Job\_Class cheap (150, 250, {L"Уборщик", L"Дворник", L"Мойщик стёкол", L"Грузчик", L"Фасовщик", L"Ремонт обуви"});

\_Job\_Class medium (251, 400, {L"Водитель", /\*L"Репетитор", L"Бухгалтер", L"Повар",\*/ L"Плотник", L"Фрезеровщик"/\*, L"Тренер"\*/});

\_Job\_Class upper\_med (401, 750, {L"Сварщик", L"Каменщик", L"Тестировщик ПО", L"Строитель", /\*L"Садовник",\*/ L"Ремонт техники", L"Джуниор-программист"/\*, L"Фотограф"\*/});

\_Job\_Class lower\_high (751, 4000, {L"Мидл-программист", L"Администратор БД", L"Шофёр", L"Ремонт автотехники", L"Эксперт ИБ"/\*, L"Журналист"\*/});

\_Job\_Class upper\_high (4001, 10000, {L"Личный охранник", L"Экономист", L"Статист", L"Тимлид", L"Пилот"});

Listing get\_random\_listing(int job\_type)

{

int category = randint(1, 5);

std::wstring name = L"";

unsigned int per\_hr = 0;

switch(category)

{

case 1:

name = cheap.random\_job\_name();

per\_hr = cheap.random\_per\_hr();

break;

case 2:

name = medium.random\_job\_name();

per\_hr = medium.random\_per\_hr();

break;

case 3:

name = upper\_med.random\_job\_name();

per\_hr = upper\_med.random\_per\_hr();

break;

case 4:

name = lower\_high.random\_job\_name();

per\_hr = lower\_high.random\_per\_hr();

break;

case 5:

name = upper\_high.random\_job\_name();

per\_hr = upper\_high.random\_per\_hr();

break;

}

/// Customer

if (job\_type == CUSTOMER\_LISTING)

{

User \*customer = PREDEFINED\_COMPANIES[randint(9)];

int hrs = randint(1, 8);

return Listing(name, hrs, per\_hr, nullptr, customer);

}

/// Contractor

else if (job\_type == CONTRACTOR\_LISTING)

{

return Listing(name, 1, per\_hr, get\_random\_individual(), nullptr);

}

}

#endif

#### Листинг кода user.hpp

#pragma once

#ifndef USER\_HPP

#define USER\_HPP

#include <vector>

#include "User/name.hpp"

#include "User/phone\_number.hpp"

#include "../utils.hpp"

/// Here be:

class User;

class Individual;

class Company;

/// Parent class for 'Individual' and 'Company'

class User

{

public:

/// POSSIBLE MEMORY LEAK: Trying to wrap it in a destructor caused errors I have no time to fix.

Name\* name;

Phone\_Number phone\_number;

Date birth\_date;

std::wstring email;

/// This is dirty, but doing something about websites available only with 'Companies' would make it even dirtier.

/// It's accessible, but useless from 'Individual' anyway.

std::wstring website;

std::wstring extra\_contacts;

User(const Date \_birth\_date, const Phone\_Number& \_phone\_number,

const std::wstring& \_email/\* = L""\*/, const std::wstring& \_extra\_contacts/\* = L""\*/)

: birth\_date(\_birth\_date), phone\_number(\_phone\_number), email(\_email), extra\_contacts(\_extra\_contacts)

{}

virtual std::wstring serialize() = 0;

};

class Individual : public User

{

public:

Individual(const Individual\_Name& \_name, const Date& \_birth\_date, const Phone\_Number& \_phone\_number,

const std::wstring& \_email = L"", const std::wstring& \_extra\_contacts = L"")

: name\_obj(\_name), User(\_birth\_date, \_phone\_number, \_email, \_extra\_contacts)

{

/// Pointer magic. Basically, 'name' used to be a pointer

/// to an object you gave the constructor. It resulted in

/// a problem where 'name' kinda existed, but kinda was in this limbo

name = &name\_obj;

}

std::wstring serialize()

{

std::wstring result =

L"individual\n"

+ birth\_date.dd\_mm\_yyyy(L" ") + L'\n'

+ name->get\_full() + L'\n'

+ phone\_number.serialize() + L'\n'

+ email + L'\n'

+ extra\_contacts + L'\n';

return result;

}

private:

Individual\_Name name\_obj;

};

class Company : public User

{

public:

Company(const Company\_Name& \_name, const Date& \_birth\_date, const Phone\_Number& \_phone\_number,

const std::wstring& \_email, const std::wstring& \_website,

const std::wstring& \_extra\_contacts = L"")

: name\_obj(\_name), User(\_birth\_date, \_phone\_number, \_email, \_extra\_contacts)

{

website = \_website;

/// Pointer magic. Basically, 'name' used to be a pointer

/// to an object you gave the constructor. It resulted in

/// a problem where 'name' kinda existed, but kinda was in this limbo

name = &name\_obj;

}

std::wstring serialize()

{

std::wstring result =

L"company\n"

+ birth\_date.dd\_mm\_yyyy(L" ") + L'\n'

+ name->get\_full() + L'\n'

+ phone\_number.serialize() + L'\n'

+ email + L'\n'

+ extra\_contacts + L'\n'

+ website + L'\n';

return result;

}

private:

Company\_Name name\_obj;

};

#endif

#### Листинг кода account\_info.hpp

#pragma once

#include "../../Classes/User.hpp"

namespace Agora {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

public ref class Account\_Info : public System::Windows::Forms::Form

{

public:

Account\_Info(User\* about, System::String^ heading, const bool contract\_made)

{

InitializeComponent();

this->Text = heading;

/// Straightforward.

name->Text = to\_dotnet\_string(about->name->get\_full());

/// Things any user type may or may not have.

if (about->email != L"")

email->Text = to\_dotnet\_string(about->email);

else

{

email->Text = L"-";

email->Enabled = false;

label\_email->Enabled = false;

}

if (about->extra\_contacts != L"")

extra\_contacts->Text = to\_dotnet\_string(about->extra\_contacts);

else

{

extra\_contacts->Text = L"-";

extra\_contacts->Enabled = false;

label\_extra\_contacts->Enabled = false;

}

if (about->website != L"")

website->Text = to\_dotnet\_string(about->website);

else

{

website->Text = L"-";

website->Enabled = false;

label\_website->Enabled = false;

}

/// User-type specifics.

if (typeid(\*about) == typeid(Individual))

{

label\_name->Text = L"Полное имя:";

if (contract\_made == true)

phone\_number->Text = to\_dotnet\_string(about->phone\_number.as\_text());

else

phone\_number->Text = to\_dotnet\_string(about->phone\_number.conceal());

birthdate->Text = to\_dotnet\_string(about->birth\_date.dd\_month\_yyyy\_genitive());

label\_birthdate->Text = L"Дата рождения:";

}

if (typeid(\*about) == typeid(Company))

{

label\_name->Text = L"Имя компании:";

phone\_number->Text = to\_dotnet\_string(about->phone\_number.as\_text());

birthdate->Text = to\_dotnet\_string(about->birth\_date.month\_yyyy\_genitive());

label\_birthdate->Text = L"На рынке с:";

}

}

#pragma region = Winforms code =

protected:

~Account\_Info()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Label^ label\_info;

private: System::Windows::Forms::PictureBox^ bg\_info;

private: System::Windows::Forms::Label^ label\_name;

private: System::Windows::Forms::Label^ label\_birthdate;

private: System::Windows::Forms::Label^ label\_phone\_number;

private: System::Windows::Forms::Label^ label\_email;

private: System::Windows::Forms::Label^ label\_extra\_contacts;

private: System::Windows::Forms::Label^ label\_website;

private: System::Windows::Forms::TextBox^ name;

private: System::Windows::Forms::TextBox^ birthdate;

private: System::Windows::Forms::TextBox^ phone\_number;

private: System::Windows::Forms::TextBox^ email;

private: System::Windows::Forms::TextBox^ extra\_contacts;

private: System::Windows::Forms::TextBox^ website;

protected:

protected:

private:

System::ComponentModel::Container ^components;

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(Account\_Info::typeid));

this->label\_info = (gcnew System::Windows::Forms::Label());

this->bg\_info = (gcnew System::Windows::Forms::PictureBox());

this->label\_name = (gcnew System::Windows::Forms::Label());

this->label\_birthdate = (gcnew System::Windows::Forms::Label());

this->label\_phone\_number = (gcnew System::Windows::Forms::Label());

this->label\_email = (gcnew System::Windows::Forms::Label());

this->label\_extra\_contacts = (gcnew System::Windows::Forms::Label());

this->label\_website = (gcnew System::Windows::Forms::Label());

this->name = (gcnew System::Windows::Forms::TextBox());

this->birthdate = (gcnew System::Windows::Forms::TextBox());

this->phone\_number = (gcnew System::Windows::Forms::TextBox());

this->email = (gcnew System::Windows::Forms::TextBox());

this->extra\_contacts = (gcnew System::Windows::Forms::TextBox());

this->website = (gcnew System::Windows::Forms::TextBox());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->bg\_info))->BeginInit();

this->SuspendLayout();

//

// label\_info

//

this->label\_info->AutoSize = true;

this->label\_info->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(232)), static\_cast<System::Int32>(static\_cast<System::Byte>(232)),

static\_cast<System::Int32>(static\_cast<System::Byte>(232)));

this->label\_info->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_info->Location = System::Drawing::Point(10, 10);

this->label\_info->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_info->Name = L"label\_info";

this->label\_info->Size = System::Drawing::Size(177, 29);

this->label\_info->TabIndex = 2;

this->label\_info->Text = L"Информация";

//

// bg\_info

//

this->bg\_info->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"bg\_info.BackgroundImage")));

this->bg\_info->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->bg\_info->Location = System::Drawing::Point(0, -5);

this->bg\_info->Margin = System::Windows::Forms::Padding(4);

this->bg\_info->Name = L"bg\_info";

this->bg\_info->Size = System::Drawing::Size(435, 176);

this->bg\_info->TabIndex = 3;

this->bg\_info->TabStop = false;

//

// label\_name

//

this->label\_name->AutoSize = true;

this->label\_name->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_name->Location = System::Drawing::Point(12, 185);

this->label\_name->Name = L"label\_name";

this->label\_name->Size = System::Drawing::Size(85, 20);

this->label\_name->TabIndex = 4;

this->label\_name->Text = L"label\_name";

//

// label\_birthdate

//

this->label\_birthdate->AutoSize = true;

this->label\_birthdate->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_birthdate->Location = System::Drawing::Point(11, 219);

this->label\_birthdate->Name = L"label\_birthdate";

this->label\_birthdate->Size = System::Drawing::Size(107, 20);

this->label\_birthdate->TabIndex = 5;

this->label\_birthdate->Text = L"label\_birthdate";

//

// label\_phone\_number

//

this->label\_phone\_number->AutoSize = true;

this->label\_phone\_number->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_phone\_number->Location = System::Drawing::Point(11, 253);

this->label\_phone\_number->Name = L"label\_phone\_number";

this->label\_phone\_number->Size = System::Drawing::Size(132, 20);

this->label\_phone\_number->TabIndex = 6;

this->label\_phone\_number->Text = L"Номер телефона:";

//

// label\_email

//

this->label\_email->AutoSize = true;

this->label\_email->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_email->Location = System::Drawing::Point(11, 287);

this->label\_email->Name = L"label\_email";

this->label\_email->Size = System::Drawing::Size(54, 20);

this->label\_email->TabIndex = 7;

this->label\_email->Text = L"e-mail:";

//

// label\_extra\_contacts

//

this->label\_extra\_contacts->AutoSize = true;

this->label\_extra\_contacts->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->label\_extra\_contacts->Location = System::Drawing::Point(12, 321);

this->label\_extra\_contacts->Name = L"label\_extra\_contacts";

this->label\_extra\_contacts->Size = System::Drawing::Size(118, 20);

this->label\_extra\_contacts->TabIndex = 8;

this->label\_extra\_contacts->Text = L"Доп. контакты:";

//

// label\_website

//

this->label\_website->AutoSize = true;

this->label\_website->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_website->Location = System::Drawing::Point(11, 355);

this->label\_website->Name = L"label\_website";

this->label\_website->Size = System::Drawing::Size(47, 20);

this->label\_website->TabIndex = 9;

this->label\_website->Text = L"Сайт:";

//

// name

//

this->name->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->name->Location = System::Drawing::Point(149, 182);

this->name->Name = L"name";

this->name->ReadOnly = true;

this->name->Size = System::Drawing::Size(272, 28);

this->name->TabIndex = 10;

//

// birthdate

//

this->birthdate->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->birthdate->Location = System::Drawing::Point(149, 216);

this->birthdate->Name = L"birthdate";

this->birthdate->ReadOnly = true;

this->birthdate->Size = System::Drawing::Size(272, 28);

this->birthdate->TabIndex = 11;

//

// phone\_number

//

this->phone\_number->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->phone\_number->Location = System::Drawing::Point(149, 250);

this->phone\_number->Name = L"phone\_number";

this->phone\_number->ReadOnly = true;

this->phone\_number->Size = System::Drawing::Size(272, 28);

this->phone\_number->TabIndex = 12;

//

// email

//

this->email->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->email->Location = System::Drawing::Point(149, 284);

this->email->Name = L"email";

this->email->ReadOnly = true;

this->email->Size = System::Drawing::Size(272, 28);

this->email->TabIndex = 13;

//

// extra\_contacts

//

this->extra\_contacts->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->extra\_contacts->Location = System::Drawing::Point(149, 318);

this->extra\_contacts->Name = L"extra\_contacts";

this->extra\_contacts->ReadOnly = true;

this->extra\_contacts->Size = System::Drawing::Size(272, 28);

this->extra\_contacts->TabIndex = 14;

//

// website

//

this->website->Font = (gcnew System::Drawing::Font(L"Roboto Condensed", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->website->Location = System::Drawing::Point(149, 352);

this->website->Name = L"website";

this->website->ReadOnly = true;

this->website->Size = System::Drawing::Size(272, 28);

this->website->TabIndex = 15;

//

// Account\_Info

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(432, 393);

this->Controls->Add(this->website);

this->Controls->Add(this->extra\_contacts);

this->Controls->Add(this->email);

this->Controls->Add(this->phone\_number);

this->Controls->Add(this->birthdate);

this->Controls->Add(this->name);

this->Controls->Add(this->label\_website);

this->Controls->Add(this->label\_extra\_contacts);

this->Controls->Add(this->label\_email);

this->Controls->Add(this->label\_phone\_number);

this->Controls->Add(this->label\_birthdate);

this->Controls->Add(this->label\_name);

this->Controls->Add(this->label\_info);

this->Controls->Add(this->bg\_info);

this->MaximumSize = System::Drawing::Size(450, 440);

this->MinimumSize = System::Drawing::Size(450, 440);

this->Name = L"Account\_Info";

this->Text = L"Об аккаунте";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->bg\_info))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

};

}

#### Листинг кода listing\_interface.hpp

#pragma once

#ifndef LISTING\_INTERFACE\_HPP

#define LISTING\_INTERFACE\_HPP

#include <Windows.h>

#include "../../Classes/Listing.hpp"

#include "../../Menus/Account\_Info/account\_info.hpp"

#include "../../Utils.hpp"

using namespace Agora;

using namespace System::Windows::Forms;

namespace Agora

{

public ref class Listing\_Interface

{

public:

GroupBox ^ui\_group;

Label ^name, ^author;

Button ^author\_info, ^button;

Label ^label\_per\_hr, ^per\_hr, ^label\_hrs;

Control ^hrs; /// May be a 'NumericUpDown' or a 'Label'

Label ^total;

Listing\* listing;

Listing\_Interface^ operator=(Listing\_Interface^ other)

{

if (this != other)

{

ui\_group = other->ui\_group;

name = other->name;

author = other->author;

author\_info = other->author\_info;

button = other->button;

label\_per\_hr = other->label\_per\_hr;

per\_hr = other->per\_hr;

label\_hrs = other->label\_hrs;

hrs = other->hrs;

total = other->total;

listing = other->listing;

}

return this;

}

/// For compiler not to whine.

Listing\_Interface()

{}

Listing\_Interface(GroupBox^ \_ui\_group, Label^ \_name, Label^ \_author,

Button^ \_author\_info, Button^ \_button,

Label^ \_label\_per\_hr, Label^ \_per\_hr, Label^ \_label\_hrs,

Control^ \_hrs, Label^ \_total)

: ui\_group(\_ui\_group), name(\_name), author(\_author),

author\_info(\_author\_info), button(\_button),

label\_per\_hr(\_label\_per\_hr), per\_hr(\_per\_hr), label\_hrs(\_label\_hrs),

hrs(\_hrs), total(\_total)

{

ui\_group->Enabled = true;

ui\_group->Visible = true;

}

/// Intended only for UI's with variable hour settings!

void calculate\_and\_display\_total(const unsigned int hrs)

{

listing->hrs = hrs;

total->Text = L"Итого, ₽: " + listing->payment\_total();

}

bool has(Control^ node)

{

/// Maybe do it via an 'unordered\_set'?

return (

node == ui\_group || node == name || node == author

|| node == author\_info || node == button

|| node == label\_per\_hr || node == per\_hr

|| node == label\_hrs || node == hrs

|| node == total

);

}

void set\_listing(Listing\* new\_listing)

{

listing = new\_listing;

ui\_group->Enabled = true;

name->Text = to\_dotnet\_string(listing->name);

per\_hr->Text = Convert::ToString(listing->per\_hr);

// NOTE: can be optimized into a mandatory 'calculate\_and\_display\_total()'

/// 'hrs' is a text label

if (Convert::ToString(hrs->GetType()) == L"System.Windows.Forms.Label")

{

hrs->Text = Convert::ToString(listing->hrs);

total->Text = L"Итого, ₽: " + listing->payment\_total();

}

/// 'hrs' is a settable variable (feed listings)

else if (Convert::ToString(hrs->GetType()) == L"System.Windows.Forms.NumericUpDown")

calculate\_and\_display\_total(1);

author->Text = to\_dotnet\_string(listing->author->name->get\_normal());

}

void show\_info()

{

System::String^ heading;

if (listing->author == listing->contractor)

heading = L"Об аккаунте подрядчика";

else if (listing->author == listing->customer)

heading = L"Об аккаунте заказчика";

bool contract\_made = listing->contractor != nullptr && listing->customer != nullptr;

Account\_Info^ account\_info = gcnew Account\_Info(listing->author, heading, contract\_made);

account\_info->ShowDialog();

}

};

}

#endif

#### Листинг кода main\_menu.hpp

#pragma once

#ifndef MAIN\_MENU\_HPP

#define MAIN\_MENU\_HPP

#include <vector>

#include "../Registration/registration.hpp"

#include "listing\_interface.hpp"

#include "../../Classes/random\_listing\_generator.hpp"

#include "../../Classes/user.hpp"

#include "../../utils.hpp"

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// 'Member of an unmanaged class' bullshit made me put all of this here

std::vector<Listing> active\_listings;

Listing listing\_contractor1, listing\_contractor2,

listing\_customer1, listing\_customer2;

const std::wstring PROGRAM\_INFO =

(std::wstring)L"Agora v.1.0 от 09.07.2025\n" // it was INSISTING I was trying to initialize with 'System::String'

+ L"Курсовой проект от Первакова А.И.\n"

+ L"КубГАУ, Прикладная информатика, ИТ2403";

namespace Agora

{

public ref class Main\_Menu : public System::Windows::Forms::Form

{

public:

User\* user;

Listing\_Interface ^contractor1, ^contractor2,

^customer1, ^customer2,

^active\_listing1, ^active\_listing2;

Main\_Menu(User\* \_user)

: user(\_user)

{

InitializeComponent();

active\_listings = load\_listings(user);

System::Drawing::Size size(665, 485);

this->Size = size;

/// I HATE CLR. FUCK CLR. MICROSOFT CAN GO EAT SHIT

System::Drawing::Font^ font = gcnew System::Drawing::Font("Roboto", 7);

this->Font = font;

/// 1. WINDOW'S scaling somewhy depends on font size. What the f\*\*\*?

/// 2. 8 shows correct size in editor; 7 makes the release program correct.

sidebar\_pick\_account(nullptr, nullptr); /// default menu may become a choice in Settings

fill\_feed\_listing\_interface();

fill\_account\_listing\_interface();

fill\_account\_menu();

fill\_feed\_menu();

}

private:

void fill\_account\_listing\_interface()

{

if (active\_listings.size() < 1)

{

active\_listing1\_ui->Visible = false;

active\_listing2\_ui->Visible = false;

return;

}

active\_listing1\_ui->Visible = true;

active\_listing1 = gcnew Listing\_Interface(

active\_listing1\_ui, active\_listing1\_name, active\_listing1\_author,

active\_listing1\_button\_info, active\_listing1\_button\_finish,

active\_listing1\_label\_per\_hr, active\_listing1\_per\_hr,

active\_listing1\_label\_hrs, active\_listing1\_hrs,

active\_listing1\_total

);

active\_listing1->set\_listing(&active\_listings[0]);

if (active\_listings.size() < 2)

{

active\_listing2\_ui->Visible = false;

return;

}

active\_listing2\_ui->Visible = true;

active\_listing2 = gcnew Listing\_Interface(

active\_listing2\_ui, active\_listing2\_name, active\_listing2\_author,

active\_listing2\_button\_info, active\_listing2\_button\_finish,

active\_listing2\_label\_per\_hr, active\_listing2\_per\_hr,

active\_listing2\_label\_hrs, active\_listing2\_hrs,

active\_listing2\_total

);

active\_listing2->set\_listing(&active\_listings[1]);

}

void fill\_feed\_listing\_interface()

{

contractor1 = gcnew Listing\_Interface(

listing\_contractor1\_ui, listing\_contractor1\_name, listing\_contractor1\_author,

listing\_contractor1\_button\_info, listing\_contractor1\_button\_hire,

listing\_contractor1\_label\_per\_hr, listing\_contractor1\_per\_hr,

listing\_contractor1\_label\_hrs, listing\_contractor1\_picker\_hrs, listing\_contractor1\_total

);

contractor2 = gcnew Listing\_Interface(

listing\_contractor2\_ui, listing\_contractor2\_name, listing\_contractor2\_author,

listing\_contractor2\_button\_info, listing\_contractor2\_button\_hire,

listing\_contractor2\_label\_per\_hr, listing\_contractor2\_per\_hr,

listing\_contractor2\_label\_hrs, listing\_contractor2\_picker\_hrs, listing\_contractor2\_total

);

customer1 = gcnew Listing\_Interface(

listing\_customer1\_ui, listing\_customer1\_name, listing\_customer1\_author,

listing\_customer1\_button\_info, listing\_customer1\_button\_accept,

listing\_customer1\_label\_per\_hr, listing\_customer1\_per\_hr,

listing\_customer1\_label\_hrs, listing\_customer1\_hrs, listing\_customer1\_total

);

customer2 = gcnew Listing\_Interface(

listing\_customer2\_ui, listing\_customer2\_name, listing\_customer2\_author,

listing\_customer2\_button\_info, listing\_customer2\_button\_accept,

listing\_customer2\_label\_per\_hr, listing\_customer2\_per\_hr,

listing\_customer2\_label\_hrs, listing\_customer2\_hrs, listing\_customer2\_total

);

}

#pragma region = Winforms Code =

protected:

~Main\_Menu()

{

if (components)

delete components;

}

private: System::Windows::Forms::GroupBox^ active\_listing2\_ui;

private: System::Windows::Forms::Button^ active\_listing2\_button\_info;

private: System::Windows::Forms::Label^ active\_listing2\_total;

private: System::Windows::Forms::Button^ active\_listing2\_button\_finish;

private: System::Windows::Forms::Label^ active\_listing2\_hrs;

private: System::Windows::Forms::Label^ active\_listing2\_label\_hrs;

private: System::Windows::Forms::Label^ active\_listing2\_author;

private: System::Windows::Forms::Label^ active\_listing2\_label\_per\_hr;

private: System::Windows::Forms::Label^ active\_listing2\_per\_hr;

private: System::Windows::Forms::Label^ active\_listing2\_name;

private: System::Windows::Forms::GroupBox^ active\_listing1\_ui;

private: System::Windows::Forms::Button^ active\_listing1\_button\_info;

private: System::Windows::Forms::Label^ active\_listing1\_total;

private: System::Windows::Forms::Button^ active\_listing1\_button\_finish;

private: System::Windows::Forms::Label^ active\_listing1\_hrs;

private: System::Windows::Forms::Label^ active\_listing1\_label\_hrs;

private: System::Windows::Forms::Label^ active\_listing1\_author;

private: System::Windows::Forms::Label^ active\_listing1\_label\_per\_hr;

private: System::Windows::Forms::Label^ active\_listing1\_per\_hr;

private: System::Windows::Forms::Label^ active\_listing1\_name;

System::Windows::Forms::Button^ menu\_button\_search;

System::Windows::Forms::Panel^ sidebar;

System::Windows::Forms::Button^ menu\_button\_info;

System::Windows::Forms::Button^ menu\_button\_settings;

System::Windows::Forms::Button^ menu\_button\_account;

System::Windows::Forms::Button^ menu\_button\_feed;

System::Windows::Forms::Panel^ menu\_account;

System::Windows::Forms::Label^ label\_my\_account;

System::Windows::Forms::PictureBox^ bg\_my\_account;

System::Windows::Forms::GroupBox^ group\_account\_data;

System::Windows::Forms::TextBox^ output\_account\_email;

System::Windows::Forms::TextBox^ output\_account\_phone\_number;

System::Windows::Forms::TextBox^ output\_account\_name;

System::Windows::Forms::TextBox^ output\_account\_extra;

private: System::Windows::Forms::Button^ button\_edit;

System::Windows::Forms::TextBox^ output\_account\_website;

System::Windows::Forms::Panel^ menu\_feed;

System::Windows::Forms::Label^ label\_feed;

System::Windows::Forms::PictureBox^ bg\_feed;

private: System::Windows::Forms::GroupBox^ group\_contractors;

private: System::Windows::Forms::GroupBox^ group\_customers;

System::Windows::Forms::Button^ button\_feed\_update;

System::Windows::Forms::Timer^ update\_timer;

System::Windows::Forms::Panel^ menu\_search;

System::Windows::Forms::Label^ label\_in\_future\_updates;

System::Windows::Forms::Label^ label\_search\_will\_be;

System::Windows::Forms::Button^ button\_search;

System::Windows::Forms::TextBox^ input\_search;

System::Windows::Forms::Label^ label\_search;

System::Windows::Forms::PictureBox^ bg\_search;

System::Windows::Forms::GroupBox^ group\_active\_contracts;

System::Windows::Forms::TextBox^ output\_account\_age;

System::Windows::Forms::Label^ label\_account\_website;

System::Windows::Forms::Label^ label\_account\_extra;

System::Windows::Forms::Label^ label\_account\_email;

System::Windows::Forms::Label^ label\_account\_age;

System::Windows::Forms::Label^ label\_account\_phone\_number;

System::Windows::Forms::RichTextBox^ label\_no\_active\_contracts;

System::Windows::Forms::Button^ button\_filter;

private: System::Windows::Forms::GroupBox^ listing\_customer1\_ui;

private: System::Windows::Forms::Label^ listing\_customer1\_name;

private: System::Windows::Forms::Label^ listing\_customer1\_author;

private: System::Windows::Forms::Label^ listing\_customer1\_label\_per\_hr;

private: System::Windows::Forms::Label^ listing\_customer1\_per\_hr;

private: System::Windows::Forms::Label^ listing\_customer1\_hrs;

private: System::Windows::Forms::Label^ listing\_customer1\_label\_hrs;

private: System::Windows::Forms::Button^ listing\_customer1\_button\_accept;

private: System::Windows::Forms::Label^ listing\_customer1\_total;

private: System::Windows::Forms::GroupBox^ listing\_customer2\_ui;

private: System::Windows::Forms::Label^ listing\_customer2\_total;

private: System::Windows::Forms::Button^ listing\_customer2\_button\_accept;

private: System::Windows::Forms::Label^ listing\_customer2\_hrs;

private: System::Windows::Forms::Label^ listing\_customer2\_label\_hrs;

private: System::Windows::Forms::Label^ listing\_customer2\_author;

private: System::Windows::Forms::Label^ listing\_customer2\_label\_per\_hr;

private: System::Windows::Forms::Label^ listing\_customer2\_per\_hr;

private: System::Windows::Forms::Label^ listing\_customer2\_name;

private: System::Windows::Forms::GroupBox^ listing\_contractor2\_ui;

private: System::Windows::Forms::Label^ listing\_contractor2\_total;

private: System::Windows::Forms::Button^ listing\_contractor2\_button\_hire;

private: System::Windows::Forms::Label^ listing\_contractor2\_label\_hrs;

private: System::Windows::Forms::Label^ listing\_contractor2\_author;

private: System::Windows::Forms::Label^ listing\_contractor2\_label\_per\_hr;

private: System::Windows::Forms::Label^ listing\_contractor2\_per\_hr;

private: System::Windows::Forms::Label^ listing\_contractor2\_name;

private: System::Windows::Forms::GroupBox^ listing\_contractor1\_ui;

private: System::Windows::Forms::Label^ listing\_contractor1\_total;

private: System::Windows::Forms::Button^ listing\_contractor1\_button\_hire;

private: System::Windows::Forms::Label^ listing\_contractor1\_label\_hrs;

private: System::Windows::Forms::Label^ listing\_contractor1\_author;

private: System::Windows::Forms::Label^ listing\_contractor1\_label\_per\_hr;

private: System::Windows::Forms::Label^ listing\_contractor1\_per\_hr;

private: System::Windows::Forms::Label^ listing\_contractor1\_name;

private: System::Windows::Forms::NumericUpDown^ listing\_contractor2\_picker\_hrs;

private: System::Windows::Forms::NumericUpDown^ listing\_contractor1\_picker\_hrs;

private: System::Windows::Forms::Button^ listing\_customer1\_button\_info;

private: System::Windows::Forms::Button^ listing\_customer2\_button\_info;

private: System::Windows::Forms::Button^ listing\_contractor2\_button\_info;

private: System::Windows::Forms::Button^ listing\_contractor1\_button\_info;

System::ComponentModel::IContainer^ components;

void InitializeComponent(void)

{

this->components = (gcnew System::ComponentModel::Container());

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(Main\_Menu::typeid));

this->menu\_button\_search = (gcnew System::Windows::Forms::Button());

this->sidebar = (gcnew System::Windows::Forms::Panel());

this->menu\_button\_info = (gcnew System::Windows::Forms::Button());

this->menu\_button\_settings = (gcnew System::Windows::Forms::Button());

this->menu\_button\_account = (gcnew System::Windows::Forms::Button());

this->menu\_button\_feed = (gcnew System::Windows::Forms::Button());

this->menu\_account = (gcnew System::Windows::Forms::Panel());

this->group\_active\_contracts = (gcnew System::Windows::Forms::GroupBox());

this->active\_listing2\_ui = (gcnew System::Windows::Forms::GroupBox());

this->active\_listing2\_button\_info = (gcnew System::Windows::Forms::Button());

this->active\_listing2\_total = (gcnew System::Windows::Forms::Label());

this->active\_listing2\_button\_finish = (gcnew System::Windows::Forms::Button());

this->active\_listing2\_hrs = (gcnew System::Windows::Forms::Label());

this->active\_listing2\_label\_hrs = (gcnew System::Windows::Forms::Label());

this->active\_listing2\_author = (gcnew System::Windows::Forms::Label());

this->active\_listing2\_label\_per\_hr = (gcnew System::Windows::Forms::Label());

this->active\_listing2\_per\_hr = (gcnew System::Windows::Forms::Label());

this->active\_listing2\_name = (gcnew System::Windows::Forms::Label());

this->active\_listing1\_ui = (gcnew System::Windows::Forms::GroupBox());

this->active\_listing1\_button\_info = (gcnew System::Windows::Forms::Button());

this->active\_listing1\_total = (gcnew System::Windows::Forms::Label());

this->active\_listing1\_button\_finish = (gcnew System::Windows::Forms::Button());

this->active\_listing1\_hrs = (gcnew System::Windows::Forms::Label());

this->active\_listing1\_label\_hrs = (gcnew System::Windows::Forms::Label());

this->active\_listing1\_author = (gcnew System::Windows::Forms::Label());

this->active\_listing1\_label\_per\_hr = (gcnew System::Windows::Forms::Label());

this->active\_listing1\_per\_hr = (gcnew System::Windows::Forms::Label());

this->active\_listing1\_name = (gcnew System::Windows::Forms::Label());

this->label\_no\_active\_contracts = (gcnew System::Windows::Forms::RichTextBox());

this->group\_account\_data = (gcnew System::Windows::Forms::GroupBox());

this->label\_account\_website = (gcnew System::Windows::Forms::Label());

this->label\_account\_extra = (gcnew System::Windows::Forms::Label());

this->label\_account\_email = (gcnew System::Windows::Forms::Label());

this->label\_account\_age = (gcnew System::Windows::Forms::Label());

this->label\_account\_phone\_number = (gcnew System::Windows::Forms::Label());

this->output\_account\_age = (gcnew System::Windows::Forms::TextBox());

this->output\_account\_website = (gcnew System::Windows::Forms::TextBox());

this->button\_edit = (gcnew System::Windows::Forms::Button());

this->output\_account\_extra = (gcnew System::Windows::Forms::TextBox());

this->output\_account\_email = (gcnew System::Windows::Forms::TextBox());

this->output\_account\_phone\_number = (gcnew System::Windows::Forms::TextBox());

this->output\_account\_name = (gcnew System::Windows::Forms::TextBox());

this->label\_my\_account = (gcnew System::Windows::Forms::Label());

this->bg\_my\_account = (gcnew System::Windows::Forms::PictureBox());

this->menu\_feed = (gcnew System::Windows::Forms::Panel());

this->button\_feed\_update = (gcnew System::Windows::Forms::Button());

this->group\_customers = (gcnew System::Windows::Forms::GroupBox());

this->listing\_customer2\_ui = (gcnew System::Windows::Forms::GroupBox());

this->listing\_customer2\_button\_info = (gcnew System::Windows::Forms::Button());

this->listing\_customer2\_total = (gcnew System::Windows::Forms::Label());

this->listing\_customer2\_button\_accept = (gcnew System::Windows::Forms::Button());

this->listing\_customer2\_hrs = (gcnew System::Windows::Forms::Label());

this->listing\_customer2\_label\_hrs = (gcnew System::Windows::Forms::Label());

this->listing\_customer2\_author = (gcnew System::Windows::Forms::Label());

this->listing\_customer2\_label\_per\_hr = (gcnew System::Windows::Forms::Label());

this->listing\_customer2\_per\_hr = (gcnew System::Windows::Forms::Label());

this->listing\_customer2\_name = (gcnew System::Windows::Forms::Label());

this->listing\_customer1\_ui = (gcnew System::Windows::Forms::GroupBox());

this->listing\_customer1\_button\_info = (gcnew System::Windows::Forms::Button());

this->listing\_customer1\_total = (gcnew System::Windows::Forms::Label());

this->listing\_customer1\_button\_accept = (gcnew System::Windows::Forms::Button());

this->listing\_customer1\_hrs = (gcnew System::Windows::Forms::Label());

this->listing\_customer1\_label\_hrs = (gcnew System::Windows::Forms::Label());

this->listing\_customer1\_author = (gcnew System::Windows::Forms::Label());

this->listing\_customer1\_label\_per\_hr = (gcnew System::Windows::Forms::Label());

this->listing\_customer1\_per\_hr = (gcnew System::Windows::Forms::Label());

this->listing\_customer1\_name = (gcnew System::Windows::Forms::Label());

this->group\_contractors = (gcnew System::Windows::Forms::GroupBox());

this->listing\_contractor2\_ui = (gcnew System::Windows::Forms::GroupBox());

this->listing\_contractor2\_button\_info = (gcnew System::Windows::Forms::Button());

this->listing\_contractor2\_picker\_hrs = (gcnew System::Windows::Forms::NumericUpDown());

this->listing\_contractor2\_total = (gcnew System::Windows::Forms::Label());

this->listing\_contractor2\_button\_hire = (gcnew System::Windows::Forms::Button());

this->listing\_contractor2\_label\_hrs = (gcnew System::Windows::Forms::Label());

this->listing\_contractor2\_author = (gcnew System::Windows::Forms::Label());

this->listing\_contractor2\_label\_per\_hr = (gcnew System::Windows::Forms::Label());

this->listing\_contractor2\_per\_hr = (gcnew System::Windows::Forms::Label());

this->listing\_contractor2\_name = (gcnew System::Windows::Forms::Label());

this->listing\_contractor1\_ui = (gcnew System::Windows::Forms::GroupBox());

this->listing\_contractor1\_button\_info = (gcnew System::Windows::Forms::Button());

this->listing\_contractor1\_picker\_hrs = (gcnew System::Windows::Forms::NumericUpDown());

this->listing\_contractor1\_total = (gcnew System::Windows::Forms::Label());

this->listing\_contractor1\_button\_hire = (gcnew System::Windows::Forms::Button());

this->listing\_contractor1\_label\_hrs = (gcnew System::Windows::Forms::Label());

this->listing\_contractor1\_author = (gcnew System::Windows::Forms::Label());

this->listing\_contractor1\_label\_per\_hr = (gcnew System::Windows::Forms::Label());

this->listing\_contractor1\_per\_hr = (gcnew System::Windows::Forms::Label());

this->listing\_contractor1\_name = (gcnew System::Windows::Forms::Label());

this->label\_feed = (gcnew System::Windows::Forms::Label());

this->bg\_feed = (gcnew System::Windows::Forms::PictureBox());

this->update\_timer = (gcnew System::Windows::Forms::Timer(this->components));

this->menu\_search = (gcnew System::Windows::Forms::Panel());

this->button\_filter = (gcnew System::Windows::Forms::Button());

this->label\_in\_future\_updates = (gcnew System::Windows::Forms::Label());

this->label\_search\_will\_be = (gcnew System::Windows::Forms::Label());

this->button\_search = (gcnew System::Windows::Forms::Button());

this->input\_search = (gcnew System::Windows::Forms::TextBox());

this->label\_search = (gcnew System::Windows::Forms::Label());

this->bg\_search = (gcnew System::Windows::Forms::PictureBox());

this->sidebar->SuspendLayout();

this->menu\_account->SuspendLayout();

this->group\_active\_contracts->SuspendLayout();

this->active\_listing2\_ui->SuspendLayout();

this->active\_listing1\_ui->SuspendLayout();

this->group\_account\_data->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->bg\_my\_account))->BeginInit();

this->menu\_feed->SuspendLayout();

this->group\_customers->SuspendLayout();

this->listing\_customer2\_ui->SuspendLayout();

this->listing\_customer1\_ui->SuspendLayout();

this->group\_contractors->SuspendLayout();

this->listing\_contractor2\_ui->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->listing\_contractor2\_picker\_hrs))->BeginInit();

this->listing\_contractor1\_ui->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->listing\_contractor1\_picker\_hrs))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->bg\_feed))->BeginInit();

this->menu\_search->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->bg\_search))->BeginInit();

this->SuspendLayout();

//

// menu\_button\_search

//

this->menu\_button\_search->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->menu\_button\_search->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Center;

this->menu\_button\_search->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"menu\_button\_search.Image")));

this->menu\_button\_search->Location = System::Drawing::Point(4, 7);

this->menu\_button\_search->Margin = System::Windows::Forms::Padding(4);

this->menu\_button\_search->Name = L"menu\_button\_search";

this->menu\_button\_search->Size = System::Drawing::Size(88, 94);

this->menu\_button\_search->TabIndex = 1;

this->menu\_button\_search->UseVisualStyleBackColor = false;

this->menu\_button\_search->Click += gcnew System::EventHandler(this, &Main\_Menu::sidebar\_pick\_search);

//

// sidebar

//

this->sidebar->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)), static\_cast<System::Int32>(static\_cast<System::Byte>(191)),

static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->sidebar->Controls->Add(this->menu\_button\_info);

this->sidebar->Controls->Add(this->menu\_button\_settings);

this->sidebar->Controls->Add(this->menu\_button\_account);

this->sidebar->Controls->Add(this->menu\_button\_feed);

this->sidebar->Controls->Add(this->menu\_button\_search);

this->sidebar->Location = System::Drawing::Point(578, 11);

this->sidebar->Margin = System::Windows::Forms::Padding(4);

this->sidebar->Name = L"sidebar";

this->sidebar->Size = System::Drawing::Size(96, 503);

this->sidebar->TabIndex = 1;

//

// menu\_button\_info

//

this->menu\_button\_info->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->menu\_button\_info->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Center;

this->menu\_button\_info->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"menu\_button\_info.Image")));

this->menu\_button\_info->Location = System::Drawing::Point(4, 401);

this->menu\_button\_info->Margin = System::Windows::Forms::Padding(4);

this->menu\_button\_info->Name = L"menu\_button\_info";

this->menu\_button\_info->Size = System::Drawing::Size(88, 94);

this->menu\_button\_info->TabIndex = 4;

this->menu\_button\_info->UseVisualStyleBackColor = false;

this->menu\_button\_info->Click += gcnew System::EventHandler(this, &Main\_Menu::sidebar\_pick\_info);

//

// menu\_button\_settings

//

this->menu\_button\_settings->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->menu\_button\_settings->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Center;

this->menu\_button\_settings->Enabled = false;

this->menu\_button\_settings->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"menu\_button\_settings.Image")));

this->menu\_button\_settings->Location = System::Drawing::Point(4, 304);

this->menu\_button\_settings->Margin = System::Windows::Forms::Padding(4);

this->menu\_button\_settings->Name = L"menu\_button\_settings";

this->menu\_button\_settings->Size = System::Drawing::Size(88, 94);

this->menu\_button\_settings->TabIndex = 3;

this->menu\_button\_settings->UseVisualStyleBackColor = false;

//

// menu\_button\_account

//

this->menu\_button\_account->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->menu\_button\_account->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Center;

this->menu\_button\_account->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"menu\_button\_account.Image")));

this->menu\_button\_account->Location = System::Drawing::Point(4, 203);

this->menu\_button\_account->Margin = System::Windows::Forms::Padding(4);

this->menu\_button\_account->Name = L"menu\_button\_account";

this->menu\_button\_account->Size = System::Drawing::Size(88, 94);

this->menu\_button\_account->TabIndex = 2;

this->menu\_button\_account->UseVisualStyleBackColor = false;

this->menu\_button\_account->Click += gcnew System::EventHandler(this, &Main\_Menu::sidebar\_pick\_account);

//

// menu\_button\_feed

//

this->menu\_button\_feed->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->menu\_button\_feed->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Center;

this->menu\_button\_feed->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"menu\_button\_feed.Image")));

this->menu\_button\_feed->Location = System::Drawing::Point(4, 104);

this->menu\_button\_feed->Margin = System::Windows::Forms::Padding(4);

this->menu\_button\_feed->Name = L"menu\_button\_feed";

this->menu\_button\_feed->Size = System::Drawing::Size(88, 94);

this->menu\_button\_feed->TabIndex = 0;

this->menu\_button\_feed->UseVisualStyleBackColor = false;

this->menu\_button\_feed->Click += gcnew System::EventHandler(this, &Main\_Menu::sidebar\_pick\_feed);

//

// menu\_account

//

this->menu\_account->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->menu\_account->Controls->Add(this->group\_active\_contracts);

this->menu\_account->Controls->Add(this->group\_account\_data);

this->menu\_account->Controls->Add(this->label\_my\_account);

this->menu\_account->Controls->Add(this->bg\_my\_account);

this->menu\_account->Location = System::Drawing::Point(10, 11);

this->menu\_account->Margin = System::Windows::Forms::Padding(4);

this->menu\_account->Name = L"menu\_account";

this->menu\_account->Size = System::Drawing::Size(560, 502);

this->menu\_account->TabIndex = 2;

//

// group\_active\_contracts

//

this->group\_active\_contracts->Controls->Add(this->active\_listing2\_ui);

this->group\_active\_contracts->Controls->Add(this->active\_listing1\_ui);

this->group\_active\_contracts->Controls->Add(this->label\_no\_active\_contracts);

this->group\_active\_contracts->Location = System::Drawing::Point(287, 168);

this->group\_active\_contracts->Margin = System::Windows::Forms::Padding(4);

this->group\_active\_contracts->Name = L"group\_active\_contracts";

this->group\_active\_contracts->Padding = System::Windows::Forms::Padding(4);

this->group\_active\_contracts->Size = System::Drawing::Size(270, 327);

this->group\_active\_contracts->TabIndex = 5;

this->group\_active\_contracts->TabStop = false;

this->group\_active\_contracts->Text = L"Активные контракты";

//

// active\_listing2\_ui

//

this->active\_listing2\_ui->Controls->Add(this->active\_listing2\_button\_info);

this->active\_listing2\_ui->Controls->Add(this->active\_listing2\_total);

this->active\_listing2\_ui->Controls->Add(this->active\_listing2\_button\_finish);

this->active\_listing2\_ui->Controls->Add(this->active\_listing2\_hrs);

this->active\_listing2\_ui->Controls->Add(this->active\_listing2\_label\_hrs);

this->active\_listing2\_ui->Controls->Add(this->active\_listing2\_author);

this->active\_listing2\_ui->Controls->Add(this->active\_listing2\_label\_per\_hr);

this->active\_listing2\_ui->Controls->Add(this->active\_listing2\_per\_hr);

this->active\_listing2\_ui->Controls->Add(this->active\_listing2\_name);

this->active\_listing2\_ui->Font = (gcnew System::Drawing::Font(L"Roboto", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing2\_ui->Location = System::Drawing::Point(8, 168);

this->active\_listing2\_ui->Margin = System::Windows::Forms::Padding(4);

this->active\_listing2\_ui->Name = L"active\_listing2\_ui";

this->active\_listing2\_ui->Padding = System::Windows::Forms::Padding(4);

this->active\_listing2\_ui->Size = System::Drawing::Size(254, 147);

this->active\_listing2\_ui->TabIndex = 10;

this->active\_listing2\_ui->TabStop = false;

//

// active\_listing2\_button\_info

//

this->active\_listing2\_button\_info->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->active\_listing2\_button\_info->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"active\_listing2\_button\_info.BackgroundImage")));

this->active\_listing2\_button\_info->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->active\_listing2\_button\_info->Location = System::Drawing::Point(5, 35);

this->active\_listing2\_button\_info->Margin = System::Windows::Forms::Padding(4);

this->active\_listing2\_button\_info->Name = L"active\_listing2\_button\_info";

this->active\_listing2\_button\_info->Size = System::Drawing::Size(30, 30);

this->active\_listing2\_button\_info->TabIndex = 5;

this->active\_listing2\_button\_info->UseVisualStyleBackColor = false;

this->active\_listing2\_button\_info->Click += gcnew System::EventHandler(this, &Main\_Menu::active\_listing\_show\_author\_info);

//

// active\_listing2\_total

//

this->active\_listing2\_total->AutoSize = true;

this->active\_listing2\_total->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing2\_total->Location = System::Drawing::Point(123, 117);

this->active\_listing2\_total->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing2\_total->Name = L"active\_listing2\_total";

this->active\_listing2\_total->Size = System::Drawing::Size(82, 20);

this->active\_listing2\_total->TabIndex = 7;

this->active\_listing2\_total->Text = L"Итого, ₽: ";

//

// active\_listing2\_button\_finish

//

this->active\_listing2\_button\_finish->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing2\_button\_finish->Location = System::Drawing::Point(5, 109);

this->active\_listing2\_button\_finish->Margin = System::Windows::Forms::Padding(4);

this->active\_listing2\_button\_finish->Name = L"active\_listing2\_button\_finish";

this->active\_listing2\_button\_finish->Size = System::Drawing::Size(119, 34);

this->active\_listing2\_button\_finish->TabIndex = 7;

this->active\_listing2\_button\_finish->Text = L"Готово";

this->active\_listing2\_button\_finish->UseVisualStyleBackColor = true;

this->active\_listing2\_button\_finish->Click += gcnew System::EventHandler(this, &Main\_Menu::finish\_active\_listing);

//

// active\_listing2\_hrs

//

this->active\_listing2\_hrs->AutoSize = true;

this->active\_listing2\_hrs->Location = System::Drawing::Point(111, 86);

this->active\_listing2\_hrs->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing2\_hrs->Name = L"active\_listing2\_hrs";

this->active\_listing2\_hrs->Size = System::Drawing::Size(126, 20);

this->active\_listing2\_hrs->TabIndex = 6;

this->active\_listing2\_hrs->Text = L"часов\_работы";

//

// active\_listing2\_label\_hrs

//

this->active\_listing2\_label\_hrs->AutoSize = true;

this->active\_listing2\_label\_hrs->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing2\_label\_hrs->Location = System::Drawing::Point(5, 86);

this->active\_listing2\_label\_hrs->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing2\_label\_hrs->Name = L"active\_listing2\_label\_hrs";

this->active\_listing2\_label\_hrs->Size = System::Drawing::Size(109, 20);

this->active\_listing2\_label\_hrs->TabIndex = 5;

this->active\_listing2\_label\_hrs->Text = L"всего часов:";

//

// active\_listing2\_author

//

this->active\_listing2\_author->AutoSize = true;

this->active\_listing2\_author->Location = System::Drawing::Point(35, 40);

this->active\_listing2\_author->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing2\_author->Name = L"active\_listing2\_author";

this->active\_listing2\_author->Size = System::Drawing::Size(134, 20);

this->active\_listing2\_author->TabIndex = 2;

this->active\_listing2\_author->Text = L"имя\_заказчика";

//

// active\_listing2\_label\_per\_hr

//

this->active\_listing2\_label\_per\_hr->AutoSize = true;

this->active\_listing2\_label\_per\_hr->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing2\_label\_per\_hr->Location = System::Drawing::Point(5, 63);

this->active\_listing2\_label\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing2\_label\_per\_hr->Name = L"active\_listing2\_label\_per\_hr";

this->active\_listing2\_label\_per\_hr->Size = System::Drawing::Size(80, 20);

this->active\_listing2\_label\_per\_hr->TabIndex = 3;

this->active\_listing2\_label\_per\_hr->Text = L"за час, ₽:";

//

// active\_listing2\_per\_hr

//

this->active\_listing2\_per\_hr->AutoSize = true;

this->active\_listing2\_per\_hr->Location = System::Drawing::Point(89, 64);

this->active\_listing2\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing2\_per\_hr->Name = L"active\_listing2\_per\_hr";

this->active\_listing2\_per\_hr->Size = System::Drawing::Size(127, 20);

this->active\_listing2\_per\_hr->TabIndex = 4;

this->active\_listing2\_per\_hr->Text = L"оплата\_за\_час";

//

// active\_listing2\_name

//

this->active\_listing2\_name->AutoSize = true;

this->active\_listing2\_name->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing2\_name->Location = System::Drawing::Point(4, 11);

this->active\_listing2\_name->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing2\_name->Name = L"active\_listing2\_name";

this->active\_listing2\_name->Size = System::Drawing::Size(170, 24);

this->active\_listing2\_name->TabIndex = 0;

this->active\_listing2\_name->Text = L"название\_заказа";

//

// active\_listing1\_ui

//

this->active\_listing1\_ui->Controls->Add(this->active\_listing1\_button\_info);

this->active\_listing1\_ui->Controls->Add(this->active\_listing1\_total);

this->active\_listing1\_ui->Controls->Add(this->active\_listing1\_button\_finish);

this->active\_listing1\_ui->Controls->Add(this->active\_listing1\_hrs);

this->active\_listing1\_ui->Controls->Add(this->active\_listing1\_label\_hrs);

this->active\_listing1\_ui->Controls->Add(this->active\_listing1\_author);

this->active\_listing1\_ui->Controls->Add(this->active\_listing1\_label\_per\_hr);

this->active\_listing1\_ui->Controls->Add(this->active\_listing1\_per\_hr);

this->active\_listing1\_ui->Controls->Add(this->active\_listing1\_name);

this->active\_listing1\_ui->Font = (gcnew System::Drawing::Font(L"Roboto", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing1\_ui->Location = System::Drawing::Point(8, 16);

this->active\_listing1\_ui->Margin = System::Windows::Forms::Padding(4);

this->active\_listing1\_ui->Name = L"active\_listing1\_ui";

this->active\_listing1\_ui->Padding = System::Windows::Forms::Padding(4);

this->active\_listing1\_ui->Size = System::Drawing::Size(254, 147);

this->active\_listing1\_ui->TabIndex = 9;

this->active\_listing1\_ui->TabStop = false;

//

// active\_listing1\_button\_info

//

this->active\_listing1\_button\_info->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->active\_listing1\_button\_info->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"active\_listing1\_button\_info.BackgroundImage")));

this->active\_listing1\_button\_info->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->active\_listing1\_button\_info->Location = System::Drawing::Point(5, 35);

this->active\_listing1\_button\_info->Margin = System::Windows::Forms::Padding(4);

this->active\_listing1\_button\_info->Name = L"active\_listing1\_button\_info";

this->active\_listing1\_button\_info->Size = System::Drawing::Size(30, 30);

this->active\_listing1\_button\_info->TabIndex = 5;

this->active\_listing1\_button\_info->UseVisualStyleBackColor = false;

this->active\_listing1\_button\_info->Click += gcnew System::EventHandler(this, &Main\_Menu::active\_listing\_show\_author\_info);

//

// active\_listing1\_total

//

this->active\_listing1\_total->AutoSize = true;

this->active\_listing1\_total->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing1\_total->Location = System::Drawing::Point(123, 117);

this->active\_listing1\_total->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing1\_total->Name = L"active\_listing1\_total";

this->active\_listing1\_total->Size = System::Drawing::Size(82, 20);

this->active\_listing1\_total->TabIndex = 7;

this->active\_listing1\_total->Text = L"Итого, ₽: ";

//

// active\_listing1\_button\_finish

//

this->active\_listing1\_button\_finish->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing1\_button\_finish->Location = System::Drawing::Point(5, 109);

this->active\_listing1\_button\_finish->Margin = System::Windows::Forms::Padding(4);

this->active\_listing1\_button\_finish->Name = L"active\_listing1\_button\_finish";

this->active\_listing1\_button\_finish->Size = System::Drawing::Size(119, 34);

this->active\_listing1\_button\_finish->TabIndex = 7;

this->active\_listing1\_button\_finish->Text = L"Готово";

this->active\_listing1\_button\_finish->UseVisualStyleBackColor = true;

this->active\_listing1\_button\_finish->Click += gcnew System::EventHandler(this, &Main\_Menu::finish\_active\_listing);

//

// active\_listing1\_hrs

//

this->active\_listing1\_hrs->AutoSize = true;

this->active\_listing1\_hrs->Location = System::Drawing::Point(111, 86);

this->active\_listing1\_hrs->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing1\_hrs->Name = L"active\_listing1\_hrs";

this->active\_listing1\_hrs->Size = System::Drawing::Size(126, 20);

this->active\_listing1\_hrs->TabIndex = 6;

this->active\_listing1\_hrs->Text = L"часов\_работы";

//

// active\_listing1\_label\_hrs

//

this->active\_listing1\_label\_hrs->AutoSize = true;

this->active\_listing1\_label\_hrs->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing1\_label\_hrs->Location = System::Drawing::Point(5, 86);

this->active\_listing1\_label\_hrs->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing1\_label\_hrs->Name = L"active\_listing1\_label\_hrs";

this->active\_listing1\_label\_hrs->Size = System::Drawing::Size(109, 20);

this->active\_listing1\_label\_hrs->TabIndex = 5;

this->active\_listing1\_label\_hrs->Text = L"всего часов:";

//

// active\_listing1\_author

//

this->active\_listing1\_author->AutoSize = true;

this->active\_listing1\_author->Location = System::Drawing::Point(35, 40);

this->active\_listing1\_author->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing1\_author->Name = L"active\_listing1\_author";

this->active\_listing1\_author->Size = System::Drawing::Size(134, 20);

this->active\_listing1\_author->TabIndex = 2;

this->active\_listing1\_author->Text = L"имя\_заказчика";

//

// active\_listing1\_label\_per\_hr

//

this->active\_listing1\_label\_per\_hr->AutoSize = true;

this->active\_listing1\_label\_per\_hr->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing1\_label\_per\_hr->Location = System::Drawing::Point(5, 63);

this->active\_listing1\_label\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing1\_label\_per\_hr->Name = L"active\_listing1\_label\_per\_hr";

this->active\_listing1\_label\_per\_hr->Size = System::Drawing::Size(80, 20);

this->active\_listing1\_label\_per\_hr->TabIndex = 3;

this->active\_listing1\_label\_per\_hr->Text = L"за час, ₽:";

//

// active\_listing1\_per\_hr

//

this->active\_listing1\_per\_hr->AutoSize = true;

this->active\_listing1\_per\_hr->Location = System::Drawing::Point(89, 64);

this->active\_listing1\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing1\_per\_hr->Name = L"active\_listing1\_per\_hr";

this->active\_listing1\_per\_hr->Size = System::Drawing::Size(127, 20);

this->active\_listing1\_per\_hr->TabIndex = 4;

this->active\_listing1\_per\_hr->Text = L"оплата\_за\_час";

//

// active\_listing1\_name

//

this->active\_listing1\_name->AutoSize = true;

this->active\_listing1\_name->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->active\_listing1\_name->Location = System::Drawing::Point(4, 11);

this->active\_listing1\_name->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->active\_listing1\_name->Name = L"active\_listing1\_name";

this->active\_listing1\_name->Size = System::Drawing::Size(170, 24);

this->active\_listing1\_name->TabIndex = 0;

this->active\_listing1\_name->Text = L"название\_заказа";

//

// label\_no\_active\_contracts

//

this->label\_no\_active\_contracts->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->label\_no\_active\_contracts->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_no\_active\_contracts->Location = System::Drawing::Point(1000, 113);

this->label\_no\_active\_contracts->Margin = System::Windows::Forms::Padding(4);

this->label\_no\_active\_contracts->Name = L"label\_no\_active\_contracts";

this->label\_no\_active\_contracts->ReadOnly = true;

this->label\_no\_active\_contracts->Size = System::Drawing::Size(246, 83);

this->label\_no\_active\_contracts->TabIndex = 0;

this->label\_no\_active\_contracts->Text = L"Активных контрактов нет.\n\nЗайдите в Ленту и выберите!";

//

// group\_account\_data

//

this->group\_account\_data->Controls->Add(this->label\_account\_website);

this->group\_account\_data->Controls->Add(this->label\_account\_extra);

this->group\_account\_data->Controls->Add(this->label\_account\_email);

this->group\_account\_data->Controls->Add(this->label\_account\_age);

this->group\_account\_data->Controls->Add(this->label\_account\_phone\_number);

this->group\_account\_data->Controls->Add(this->output\_account\_age);

this->group\_account\_data->Controls->Add(this->output\_account\_website);

this->group\_account\_data->Controls->Add(this->button\_edit);

this->group\_account\_data->Controls->Add(this->output\_account\_extra);

this->group\_account\_data->Controls->Add(this->output\_account\_email);

this->group\_account\_data->Controls->Add(this->output\_account\_phone\_number);

this->group\_account\_data->Controls->Add(this->output\_account\_name);

this->group\_account\_data->Location = System::Drawing::Point(7, 168);

this->group\_account\_data->Margin = System::Windows::Forms::Padding(4);

this->group\_account\_data->Name = L"group\_account\_data";

this->group\_account\_data->Padding = System::Windows::Forms::Padding(4);

this->group\_account\_data->Size = System::Drawing::Size(275, 327);

this->group\_account\_data->TabIndex = 2;

this->group\_account\_data->TabStop = false;

this->group\_account\_data->Text = L"Данные";

//

// label\_account\_website

//

this->label\_account\_website->AutoSize = true;

this->label\_account\_website->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_account\_website->Location = System::Drawing::Point(5, 173);

this->label\_account\_website->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_account\_website->Name = L"label\_account\_website";

this->label\_account\_website->Size = System::Drawing::Size(53, 20);

this->label\_account\_website->TabIndex = 9;

this->label\_account\_website->Text = L"Сайт:";

//

// label\_account\_extra

//

this->label\_account\_extra->AutoSize = true;

this->label\_account\_extra->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_account\_extra->Location = System::Drawing::Point(5, 143);

this->label\_account\_extra->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_account\_extra->Name = L"label\_account\_extra";

this->label\_account\_extra->Size = System::Drawing::Size(52, 20);

this->label\_account\_extra->TabIndex = 8;

this->label\_account\_extra->Text = L"Доп.:";

//

// label\_account\_email

//

this->label\_account\_email->AutoSize = true;

this->label\_account\_email->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_account\_email->Location = System::Drawing::Point(5, 113);

this->label\_account\_email->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_account\_email->Name = L"label\_account\_email";

this->label\_account\_email->Size = System::Drawing::Size(64, 20);

this->label\_account\_email->TabIndex = 7;

this->label\_account\_email->Text = L"e-mail:";

//

// label\_account\_age

//

this->label\_account\_age->AutoSize = true;

this->label\_account\_age->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_account\_age->Location = System::Drawing::Point(5, 53);

this->label\_account\_age->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_account\_age->Name = L"label\_account\_age";

this->label\_account\_age->Size = System::Drawing::Size(83, 20);

this->label\_account\_age->TabIndex = 6;

this->label\_account\_age->Text = L"label\_age";

//

// label\_account\_phone\_number

//

this->label\_account\_phone\_number->AutoSize = true;

this->label\_account\_phone\_number->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_account\_phone\_number->Location = System::Drawing::Point(5, 83);

this->label\_account\_phone\_number->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_account\_phone\_number->Name = L"label\_account\_phone\_number";

this->label\_account\_phone\_number->Size = System::Drawing::Size(84, 20);

this->label\_account\_phone\_number->TabIndex = 6;

this->label\_account\_phone\_number->Text = L"Телефон:";

//

// output\_account\_age

//

this->output\_account\_age->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->output\_account\_age->Location = System::Drawing::Point(115, 50);

this->output\_account\_age->Margin = System::Windows::Forms::Padding(4);

this->output\_account\_age->Name = L"output\_account\_age";

this->output\_account\_age->ReadOnly = true;

this->output\_account\_age->Size = System::Drawing::Size(154, 28);

this->output\_account\_age->TabIndex = 5;

//

// output\_account\_website

//

this->output\_account\_website->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->output\_account\_website->Location = System::Drawing::Point(69, 172);

this->output\_account\_website->Margin = System::Windows::Forms::Padding(4);

this->output\_account\_website->Name = L"output\_account\_website";

this->output\_account\_website->ReadOnly = true;

this->output\_account\_website->Size = System::Drawing::Size(201, 28);

this->output\_account\_website->TabIndex = 4;

//

// button\_edit

//

this->button\_edit->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)), static\_cast<System::Int32>(static\_cast<System::Byte>(191)),

static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->button\_edit->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button\_edit->Location = System::Drawing::Point(0, 202);

this->button\_edit->Margin = System::Windows::Forms::Padding(4);

this->button\_edit->Name = L"button\_edit";

this->button\_edit->Size = System::Drawing::Size(270, 37);

this->button\_edit->TabIndex = 3;

this->button\_edit->Text = L"Редактировать";

this->button\_edit->UseVisualStyleBackColor = false;

this->button\_edit->Click += gcnew System::EventHandler(this, &Main\_Menu::edit\_account\_data);

//

// output\_account\_extra

//

this->output\_account\_extra->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->output\_account\_extra->Location = System::Drawing::Point(69, 142);

this->output\_account\_extra->Margin = System::Windows::Forms::Padding(4);

this->output\_account\_extra->Name = L"output\_account\_extra";

this->output\_account\_extra->ReadOnly = true;

this->output\_account\_extra->Size = System::Drawing::Size(201, 28);

this->output\_account\_extra->TabIndex = 3;

//

// output\_account\_email

//

this->output\_account\_email->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->output\_account\_email->Location = System::Drawing::Point(69, 112);

this->output\_account\_email->Margin = System::Windows::Forms::Padding(4);

this->output\_account\_email->Name = L"output\_account\_email";

this->output\_account\_email->ReadOnly = true;

this->output\_account\_email->Size = System::Drawing::Size(201, 28);

this->output\_account\_email->TabIndex = 2;

//

// output\_account\_phone\_number

//

this->output\_account\_phone\_number->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->output\_account\_phone\_number->Location = System::Drawing::Point(115, 80);

this->output\_account\_phone\_number->Margin = System::Windows::Forms::Padding(4);

this->output\_account\_phone\_number->Name = L"output\_account\_phone\_number";

this->output\_account\_phone\_number->ReadOnly = true;

this->output\_account\_phone\_number->Size = System::Drawing::Size(154, 28);

this->output\_account\_phone\_number->TabIndex = 1;

//

// output\_account\_name

//

this->output\_account\_name->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->output\_account\_name->Location = System::Drawing::Point(5, 20);

this->output\_account\_name->Margin = System::Windows::Forms::Padding(4);

this->output\_account\_name->Name = L"output\_account\_name";

this->output\_account\_name->ReadOnly = true;

this->output\_account\_name->Size = System::Drawing::Size(265, 28);

this->output\_account\_name->TabIndex = 0;

//

// label\_my\_account

//

this->label\_my\_account->AutoSize = true;

this->label\_my\_account->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(232)),

static\_cast<System::Int32>(static\_cast<System::Byte>(232)), static\_cast<System::Int32>(static\_cast<System::Byte>(232)));

this->label\_my\_account->Font = (gcnew System::Drawing::Font(L"Roboto", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_my\_account->Location = System::Drawing::Point(7, 7);

this->label\_my\_account->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_my\_account->Name = L"label\_my\_account";

this->label\_my\_account->Size = System::Drawing::Size(151, 28);

this->label\_my\_account->TabIndex = 0;

this->label\_my\_account->Text = L"Мой аккаунт";

//

// bg\_my\_account

//

this->bg\_my\_account->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"bg\_my\_account.BackgroundImage")));

this->bg\_my\_account->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->bg\_my\_account->Location = System::Drawing::Point(0, -7);

this->bg\_my\_account->Margin = System::Windows::Forms::Padding(4);

this->bg\_my\_account->Name = L"bg\_my\_account";

this->bg\_my\_account->Size = System::Drawing::Size(560, 169);

this->bg\_my\_account->TabIndex = 1;

this->bg\_my\_account->TabStop = false;

//

// menu\_feed

//

this->menu\_feed->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->menu\_feed->Controls->Add(this->button\_feed\_update);

this->menu\_feed->Controls->Add(this->group\_customers);

this->menu\_feed->Controls->Add(this->group\_contractors);

this->menu\_feed->Controls->Add(this->label\_feed);

this->menu\_feed->Controls->Add(this->bg\_feed);

this->menu\_feed->Location = System::Drawing::Point(10, 11);

this->menu\_feed->Margin = System::Windows::Forms::Padding(4);

this->menu\_feed->Name = L"menu\_feed";

this->menu\_feed->Size = System::Drawing::Size(560, 502);

this->menu\_feed->TabIndex = 3;

//

// button\_feed\_update

//

this->button\_feed\_update->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"button\_feed\_update.BackgroundImage")));

this->button\_feed\_update->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Center;

this->button\_feed\_update->Location = System::Drawing::Point(231, 4);

this->button\_feed\_update->Margin = System::Windows::Forms::Padding(4);

this->button\_feed\_update->Name = L"button\_feed\_update";

this->button\_feed\_update->RightToLeft = System::Windows::Forms::RightToLeft::No;

this->button\_feed\_update->Size = System::Drawing::Size(35, 37);

this->button\_feed\_update->TabIndex = 0;

this->button\_feed\_update->UseVisualStyleBackColor = true;

this->button\_feed\_update->Click += gcnew System::EventHandler(this, &Main\_Menu::update\_feed);

//

// group\_customers

//

this->group\_customers->Controls->Add(this->listing\_customer2\_ui);

this->group\_customers->Controls->Add(this->listing\_customer1\_ui);

this->group\_customers->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->group\_customers->Location = System::Drawing::Point(282, 168);

this->group\_customers->Margin = System::Windows::Forms::Padding(4);

this->group\_customers->Name = L"group\_customers";

this->group\_customers->Padding = System::Windows::Forms::Padding(4);

this->group\_customers->Size = System::Drawing::Size(270, 323);

this->group\_customers->TabIndex = 6;

this->group\_customers->TabStop = false;

this->group\_customers->Text = L"Ищут подрядчика";

//

// listing\_customer2\_ui

//

this->listing\_customer2\_ui->Controls->Add(this->listing\_customer2\_button\_info);

this->listing\_customer2\_ui->Controls->Add(this->listing\_customer2\_total);

this->listing\_customer2\_ui->Controls->Add(this->listing\_customer2\_button\_accept);

this->listing\_customer2\_ui->Controls->Add(this->listing\_customer2\_hrs);

this->listing\_customer2\_ui->Controls->Add(this->listing\_customer2\_label\_hrs);

this->listing\_customer2\_ui->Controls->Add(this->listing\_customer2\_author);

this->listing\_customer2\_ui->Controls->Add(this->listing\_customer2\_label\_per\_hr);

this->listing\_customer2\_ui->Controls->Add(this->listing\_customer2\_per\_hr);

this->listing\_customer2\_ui->Controls->Add(this->listing\_customer2\_name);

this->listing\_customer2\_ui->Font = (gcnew System::Drawing::Font(L"Roboto", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_customer2\_ui->Location = System::Drawing::Point(8, 168);

this->listing\_customer2\_ui->Margin = System::Windows::Forms::Padding(4);

this->listing\_customer2\_ui->Name = L"listing\_customer2\_ui";

this->listing\_customer2\_ui->Padding = System::Windows::Forms::Padding(4);

this->listing\_customer2\_ui->Size = System::Drawing::Size(254, 147);

this->listing\_customer2\_ui->TabIndex = 8;

this->listing\_customer2\_ui->TabStop = false;

//

// listing\_customer2\_button\_info

//

this->listing\_customer2\_button\_info->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->listing\_customer2\_button\_info->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"listing\_customer2\_button\_info.BackgroundImage")));

this->listing\_customer2\_button\_info->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->listing\_customer2\_button\_info->Location = System::Drawing::Point(5, 35);

this->listing\_customer2\_button\_info->Margin = System::Windows::Forms::Padding(4);

this->listing\_customer2\_button\_info->Name = L"listing\_customer2\_button\_info";

this->listing\_customer2\_button\_info->Size = System::Drawing::Size(30, 30);

this->listing\_customer2\_button\_info->TabIndex = 8;

this->listing\_customer2\_button\_info->UseVisualStyleBackColor = false;

this->listing\_customer2\_button\_info->Click += gcnew System::EventHandler(this, &Main\_Menu::listing\_show\_author\_info);

//

// listing\_customer2\_total

//

this->listing\_customer2\_total->AutoSize = true;

this->listing\_customer2\_total->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_customer2\_total->Location = System::Drawing::Point(123, 117);

this->listing\_customer2\_total->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer2\_total->Name = L"listing\_customer2\_total";

this->listing\_customer2\_total->Size = System::Drawing::Size(82, 20);

this->listing\_customer2\_total->TabIndex = 7;

this->listing\_customer2\_total->Text = L"Итого, ₽: ";

//

// listing\_customer2\_button\_accept

//

this->listing\_customer2\_button\_accept->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_customer2\_button\_accept->Location = System::Drawing::Point(5, 109);

this->listing\_customer2\_button\_accept->Margin = System::Windows::Forms::Padding(4);

this->listing\_customer2\_button\_accept->Name = L"listing\_customer2\_button\_accept";

this->listing\_customer2\_button\_accept->Size = System::Drawing::Size(119, 34);

this->listing\_customer2\_button\_accept->TabIndex = 7;

this->listing\_customer2\_button\_accept->Text = L"За работу!";

this->listing\_customer2\_button\_accept->UseVisualStyleBackColor = true;

this->listing\_customer2\_button\_accept->Click += gcnew System::EventHandler(this, &Main\_Menu::accept\_listing);

//

// listing\_customer2\_hrs

//

this->listing\_customer2\_hrs->AutoSize = true;

this->listing\_customer2\_hrs->Location = System::Drawing::Point(111, 86);

this->listing\_customer2\_hrs->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer2\_hrs->Name = L"listing\_customer2\_hrs";

this->listing\_customer2\_hrs->Size = System::Drawing::Size(126, 20);

this->listing\_customer2\_hrs->TabIndex = 6;

this->listing\_customer2\_hrs->Text = L"часов\_работы";

//

// listing\_customer2\_label\_hrs

//

this->listing\_customer2\_label\_hrs->AutoSize = true;

this->listing\_customer2\_label\_hrs->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_customer2\_label\_hrs->Location = System::Drawing::Point(5, 86);

this->listing\_customer2\_label\_hrs->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer2\_label\_hrs->Name = L"listing\_customer2\_label\_hrs";

this->listing\_customer2\_label\_hrs->Size = System::Drawing::Size(109, 20);

this->listing\_customer2\_label\_hrs->TabIndex = 5;

this->listing\_customer2\_label\_hrs->Text = L"всего часов:";

//

// listing\_customer2\_author

//

this->listing\_customer2\_author->AutoSize = true;

this->listing\_customer2\_author->Location = System::Drawing::Point(35, 40);

this->listing\_customer2\_author->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer2\_author->Name = L"listing\_customer2\_author";

this->listing\_customer2\_author->Size = System::Drawing::Size(134, 20);

this->listing\_customer2\_author->TabIndex = 2;

this->listing\_customer2\_author->Text = L"имя\_заказчика";

//

// listing\_customer2\_label\_per\_hr

//

this->listing\_customer2\_label\_per\_hr->AutoSize = true;

this->listing\_customer2\_label\_per\_hr->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->listing\_customer2\_label\_per\_hr->Location = System::Drawing::Point(5, 63);

this->listing\_customer2\_label\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer2\_label\_per\_hr->Name = L"listing\_customer2\_label\_per\_hr";

this->listing\_customer2\_label\_per\_hr->Size = System::Drawing::Size(80, 20);

this->listing\_customer2\_label\_per\_hr->TabIndex = 3;

this->listing\_customer2\_label\_per\_hr->Text = L"за час, ₽:";

//

// listing\_customer2\_per\_hr

//

this->listing\_customer2\_per\_hr->AutoSize = true;

this->listing\_customer2\_per\_hr->Location = System::Drawing::Point(89, 64);

this->listing\_customer2\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer2\_per\_hr->Name = L"listing\_customer2\_per\_hr";

this->listing\_customer2\_per\_hr->Size = System::Drawing::Size(127, 20);

this->listing\_customer2\_per\_hr->TabIndex = 4;

this->listing\_customer2\_per\_hr->Text = L"оплата\_за\_час";

//

// listing\_customer2\_name

//

this->listing\_customer2\_name->AutoSize = true;

this->listing\_customer2\_name->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_customer2\_name->Location = System::Drawing::Point(4, 11);

this->listing\_customer2\_name->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer2\_name->Name = L"listing\_customer2\_name";

this->listing\_customer2\_name->Size = System::Drawing::Size(170, 24);

this->listing\_customer2\_name->TabIndex = 0;

this->listing\_customer2\_name->Text = L"название\_заказа";

//

// listing\_customer1\_ui

//

this->listing\_customer1\_ui->Controls->Add(this->listing\_customer1\_button\_info);

this->listing\_customer1\_ui->Controls->Add(this->listing\_customer1\_total);

this->listing\_customer1\_ui->Controls->Add(this->listing\_customer1\_button\_accept);

this->listing\_customer1\_ui->Controls->Add(this->listing\_customer1\_hrs);

this->listing\_customer1\_ui->Controls->Add(this->listing\_customer1\_label\_hrs);

this->listing\_customer1\_ui->Controls->Add(this->listing\_customer1\_author);

this->listing\_customer1\_ui->Controls->Add(this->listing\_customer1\_label\_per\_hr);

this->listing\_customer1\_ui->Controls->Add(this->listing\_customer1\_per\_hr);

this->listing\_customer1\_ui->Controls->Add(this->listing\_customer1\_name);

this->listing\_customer1\_ui->Font = (gcnew System::Drawing::Font(L"Roboto", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_customer1\_ui->Location = System::Drawing::Point(8, 20);

this->listing\_customer1\_ui->Margin = System::Windows::Forms::Padding(4);

this->listing\_customer1\_ui->Name = L"listing\_customer1\_ui";

this->listing\_customer1\_ui->Padding = System::Windows::Forms::Padding(4);

this->listing\_customer1\_ui->Size = System::Drawing::Size(254, 147);

this->listing\_customer1\_ui->TabIndex = 0;

this->listing\_customer1\_ui->TabStop = false;

//

// listing\_customer1\_button\_info

//

this->listing\_customer1\_button\_info->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->listing\_customer1\_button\_info->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"listing\_customer1\_button\_info.BackgroundImage")));

this->listing\_customer1\_button\_info->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->listing\_customer1\_button\_info->Location = System::Drawing::Point(5, 35);

this->listing\_customer1\_button\_info->Margin = System::Windows::Forms::Padding(4);

this->listing\_customer1\_button\_info->Name = L"listing\_customer1\_button\_info";

this->listing\_customer1\_button\_info->Size = System::Drawing::Size(30, 30);

this->listing\_customer1\_button\_info->TabIndex = 5;

this->listing\_customer1\_button\_info->UseVisualStyleBackColor = false;

this->listing\_customer1\_button\_info->Click += gcnew System::EventHandler(this, &Main\_Menu::listing\_show\_author\_info);

//

// listing\_customer1\_total

//

this->listing\_customer1\_total->AutoSize = true;

this->listing\_customer1\_total->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_customer1\_total->Location = System::Drawing::Point(123, 117);

this->listing\_customer1\_total->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer1\_total->Name = L"listing\_customer1\_total";

this->listing\_customer1\_total->Size = System::Drawing::Size(82, 20);

this->listing\_customer1\_total->TabIndex = 7;

this->listing\_customer1\_total->Text = L"Итого, ₽: ";

//

// listing\_customer1\_button\_accept

//

this->listing\_customer1\_button\_accept->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_customer1\_button\_accept->Location = System::Drawing::Point(5, 109);

this->listing\_customer1\_button\_accept->Margin = System::Windows::Forms::Padding(4);

this->listing\_customer1\_button\_accept->Name = L"listing\_customer1\_button\_accept";

this->listing\_customer1\_button\_accept->Size = System::Drawing::Size(119, 34);

this->listing\_customer1\_button\_accept->TabIndex = 7;

this->listing\_customer1\_button\_accept->Text = L"За работу!";

this->listing\_customer1\_button\_accept->UseVisualStyleBackColor = true;

this->listing\_customer1\_button\_accept->Click += gcnew System::EventHandler(this, &Main\_Menu::accept\_listing);

//

// listing\_customer1\_hrs

//

this->listing\_customer1\_hrs->AutoSize = true;

this->listing\_customer1\_hrs->Location = System::Drawing::Point(111, 86);

this->listing\_customer1\_hrs->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer1\_hrs->Name = L"listing\_customer1\_hrs";

this->listing\_customer1\_hrs->Size = System::Drawing::Size(126, 20);

this->listing\_customer1\_hrs->TabIndex = 6;

this->listing\_customer1\_hrs->Text = L"часов\_работы";

//

// listing\_customer1\_label\_hrs

//

this->listing\_customer1\_label\_hrs->AutoSize = true;

this->listing\_customer1\_label\_hrs->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_customer1\_label\_hrs->Location = System::Drawing::Point(5, 86);

this->listing\_customer1\_label\_hrs->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer1\_label\_hrs->Name = L"listing\_customer1\_label\_hrs";

this->listing\_customer1\_label\_hrs->Size = System::Drawing::Size(109, 20);

this->listing\_customer1\_label\_hrs->TabIndex = 5;

this->listing\_customer1\_label\_hrs->Text = L"всего часов:";

//

// listing\_customer1\_author

//

this->listing\_customer1\_author->AutoSize = true;

this->listing\_customer1\_author->Location = System::Drawing::Point(35, 40);

this->listing\_customer1\_author->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer1\_author->Name = L"listing\_customer1\_author";

this->listing\_customer1\_author->Size = System::Drawing::Size(134, 20);

this->listing\_customer1\_author->TabIndex = 2;

this->listing\_customer1\_author->Text = L"имя\_заказчика";

//

// listing\_customer1\_label\_per\_hr

//

this->listing\_customer1\_label\_per\_hr->AutoSize = true;

this->listing\_customer1\_label\_per\_hr->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->listing\_customer1\_label\_per\_hr->Location = System::Drawing::Point(5, 63);

this->listing\_customer1\_label\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer1\_label\_per\_hr->Name = L"listing\_customer1\_label\_per\_hr";

this->listing\_customer1\_label\_per\_hr->Size = System::Drawing::Size(80, 20);

this->listing\_customer1\_label\_per\_hr->TabIndex = 3;

this->listing\_customer1\_label\_per\_hr->Text = L"за час, ₽:";

//

// listing\_customer1\_per\_hr

//

this->listing\_customer1\_per\_hr->AutoSize = true;

this->listing\_customer1\_per\_hr->Location = System::Drawing::Point(89, 64);

this->listing\_customer1\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer1\_per\_hr->Name = L"listing\_customer1\_per\_hr";

this->listing\_customer1\_per\_hr->Size = System::Drawing::Size(127, 20);

this->listing\_customer1\_per\_hr->TabIndex = 4;

this->listing\_customer1\_per\_hr->Text = L"оплата\_за\_час";

//

// listing\_customer1\_name

//

this->listing\_customer1\_name->AutoSize = true;

this->listing\_customer1\_name->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_customer1\_name->Location = System::Drawing::Point(4, 11);

this->listing\_customer1\_name->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_customer1\_name->Name = L"listing\_customer1\_name";

this->listing\_customer1\_name->Size = System::Drawing::Size(170, 24);

this->listing\_customer1\_name->TabIndex = 0;

this->listing\_customer1\_name->Text = L"название\_заказа";

//

// group\_contractors

//

this->group\_contractors->Controls->Add(this->listing\_contractor2\_ui);

this->group\_contractors->Controls->Add(this->listing\_contractor1\_ui);

this->group\_contractors->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->group\_contractors->Location = System::Drawing::Point(7, 168);

this->group\_contractors->Margin = System::Windows::Forms::Padding(4);

this->group\_contractors->Name = L"group\_contractors";

this->group\_contractors->Padding = System::Windows::Forms::Padding(4);

this->group\_contractors->Size = System::Drawing::Size(270, 323);

this->group\_contractors->TabIndex = 5;

this->group\_contractors->TabStop = false;

this->group\_contractors->Text = L"Ищут заказчика";

//

// listing\_contractor2\_ui

//

this->listing\_contractor2\_ui->Controls->Add(this->listing\_contractor2\_button\_info);

this->listing\_contractor2\_ui->Controls->Add(this->listing\_contractor2\_picker\_hrs);

this->listing\_contractor2\_ui->Controls->Add(this->listing\_contractor2\_total);

this->listing\_contractor2\_ui->Controls->Add(this->listing\_contractor2\_button\_hire);

this->listing\_contractor2\_ui->Controls->Add(this->listing\_contractor2\_label\_hrs);

this->listing\_contractor2\_ui->Controls->Add(this->listing\_contractor2\_author);

this->listing\_contractor2\_ui->Controls->Add(this->listing\_contractor2\_label\_per\_hr);

this->listing\_contractor2\_ui->Controls->Add(this->listing\_contractor2\_per\_hr);

this->listing\_contractor2\_ui->Controls->Add(this->listing\_contractor2\_name);

this->listing\_contractor2\_ui->Font = (gcnew System::Drawing::Font(L"Roboto", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_contractor2\_ui->Location = System::Drawing::Point(8, 168);

this->listing\_contractor2\_ui->Margin = System::Windows::Forms::Padding(4);

this->listing\_contractor2\_ui->Name = L"listing\_contractor2\_ui";

this->listing\_contractor2\_ui->Padding = System::Windows::Forms::Padding(4);

this->listing\_contractor2\_ui->Size = System::Drawing::Size(254, 147);

this->listing\_contractor2\_ui->TabIndex = 9;

this->listing\_contractor2\_ui->TabStop = false;

//

// listing\_contractor2\_button\_info

//

this->listing\_contractor2\_button\_info->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->listing\_contractor2\_button\_info->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"listing\_contractor2\_button\_info.BackgroundImage")));

this->listing\_contractor2\_button\_info->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->listing\_contractor2\_button\_info->Location = System::Drawing::Point(5, 35);

this->listing\_contractor2\_button\_info->Margin = System::Windows::Forms::Padding(4);

this->listing\_contractor2\_button\_info->Name = L"listing\_contractor2\_button\_info";

this->listing\_contractor2\_button\_info->Size = System::Drawing::Size(30, 30);

this->listing\_contractor2\_button\_info->TabIndex = 10;

this->listing\_contractor2\_button\_info->UseVisualStyleBackColor = false;

this->listing\_contractor2\_button\_info->Click += gcnew System::EventHandler(this, &Main\_Menu::listing\_show\_author\_info);

//

// listing\_contractor2\_picker\_hrs

//

this->listing\_contractor2\_picker\_hrs->Location = System::Drawing::Point(114, 84);

this->listing\_contractor2\_picker\_hrs->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 8, 0, 0, 0 });

this->listing\_contractor2\_picker\_hrs->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0, 0, 0 });

this->listing\_contractor2\_picker\_hrs->Name = L"listing\_contractor2\_picker\_hrs";

this->listing\_contractor2\_picker\_hrs->Size = System::Drawing::Size(40, 28);

this->listing\_contractor2\_picker\_hrs->TabIndex = 10;

this->listing\_contractor2\_picker\_hrs->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0, 0, 0 });

this->listing\_contractor2\_picker\_hrs->ValueChanged += gcnew System::EventHandler(this, &Main\_Menu::feed\_contracts\_hrs\_adjusted);

//

// listing\_contractor2\_total

//

this->listing\_contractor2\_total->AutoSize = true;

this->listing\_contractor2\_total->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_contractor2\_total->Location = System::Drawing::Point(123, 117);

this->listing\_contractor2\_total->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor2\_total->Name = L"listing\_contractor2\_total";

this->listing\_contractor2\_total->Size = System::Drawing::Size(82, 20);

this->listing\_contractor2\_total->TabIndex = 7;

this->listing\_contractor2\_total->Text = L"Итого, ₽: ";

//

// listing\_contractor2\_button\_hire

//

this->listing\_contractor2\_button\_hire->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_contractor2\_button\_hire->Location = System::Drawing::Point(5, 109);

this->listing\_contractor2\_button\_hire->Margin = System::Windows::Forms::Padding(4);

this->listing\_contractor2\_button\_hire->Name = L"listing\_contractor2\_button\_hire";

this->listing\_contractor2\_button\_hire->Size = System::Drawing::Size(119, 34);

this->listing\_contractor2\_button\_hire->TabIndex = 7;

this->listing\_contractor2\_button\_hire->Text = L"Нанять!";

this->listing\_contractor2\_button\_hire->UseVisualStyleBackColor = true;

this->listing\_contractor2\_button\_hire->Click += gcnew System::EventHandler(this, &Main\_Menu::accept\_listing);

//

// listing\_contractor2\_label\_hrs

//

this->listing\_contractor2\_label\_hrs->AutoSize = true;

this->listing\_contractor2\_label\_hrs->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_contractor2\_label\_hrs->Location = System::Drawing::Point(5, 86);

this->listing\_contractor2\_label\_hrs->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor2\_label\_hrs->Name = L"listing\_contractor2\_label\_hrs";

this->listing\_contractor2\_label\_hrs->Size = System::Drawing::Size(109, 20);

this->listing\_contractor2\_label\_hrs->TabIndex = 5;

this->listing\_contractor2\_label\_hrs->Text = L"всего часов:";

//

// listing\_contractor2\_author

//

this->listing\_contractor2\_author->AutoSize = true;

this->listing\_contractor2\_author->Location = System::Drawing::Point(35, 40);

this->listing\_contractor2\_author->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor2\_author->Name = L"listing\_contractor2\_author";

this->listing\_contractor2\_author->Size = System::Drawing::Size(149, 20);

this->listing\_contractor2\_author->TabIndex = 2;

this->listing\_contractor2\_author->Text = L"имя\_подрядчика";

//

// listing\_contractor2\_label\_per\_hr

//

this->listing\_contractor2\_label\_per\_hr->AutoSize = true;

this->listing\_contractor2\_label\_per\_hr->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->listing\_contractor2\_label\_per\_hr->Location = System::Drawing::Point(5, 63);

this->listing\_contractor2\_label\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor2\_label\_per\_hr->Name = L"listing\_contractor2\_label\_per\_hr";

this->listing\_contractor2\_label\_per\_hr->Size = System::Drawing::Size(80, 20);

this->listing\_contractor2\_label\_per\_hr->TabIndex = 3;

this->listing\_contractor2\_label\_per\_hr->Text = L"за час, ₽:";

//

// listing\_contractor2\_per\_hr

//

this->listing\_contractor2\_per\_hr->AutoSize = true;

this->listing\_contractor2\_per\_hr->Location = System::Drawing::Point(89, 64);

this->listing\_contractor2\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor2\_per\_hr->Name = L"listing\_contractor2\_per\_hr";

this->listing\_contractor2\_per\_hr->Size = System::Drawing::Size(117, 20);

this->listing\_contractor2\_per\_hr->TabIndex = 4;

this->listing\_contractor2\_per\_hr->Text = L"плата\_за\_час";

//

// listing\_contractor2\_name

//

this->listing\_contractor2\_name->AutoSize = true;

this->listing\_contractor2\_name->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_contractor2\_name->Location = System::Drawing::Point(4, 11);

this->listing\_contractor2\_name->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor2\_name->Name = L"listing\_contractor2\_name";

this->listing\_contractor2\_name->Size = System::Drawing::Size(186, 24);

this->listing\_contractor2\_name->TabIndex = 0;

this->listing\_contractor2\_name->Text = L"название\_подряда";

//

// listing\_contractor1\_ui

//

this->listing\_contractor1\_ui->Controls->Add(this->listing\_contractor1\_button\_info);

this->listing\_contractor1\_ui->Controls->Add(this->listing\_contractor1\_picker\_hrs);

this->listing\_contractor1\_ui->Controls->Add(this->listing\_contractor1\_total);

this->listing\_contractor1\_ui->Controls->Add(this->listing\_contractor1\_button\_hire);

this->listing\_contractor1\_ui->Controls->Add(this->listing\_contractor1\_label\_hrs);

this->listing\_contractor1\_ui->Controls->Add(this->listing\_contractor1\_author);

this->listing\_contractor1\_ui->Controls->Add(this->listing\_contractor1\_label\_per\_hr);

this->listing\_contractor1\_ui->Controls->Add(this->listing\_contractor1\_per\_hr);

this->listing\_contractor1\_ui->Controls->Add(this->listing\_contractor1\_name);

this->listing\_contractor1\_ui->Font = (gcnew System::Drawing::Font(L"Roboto", 10, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_contractor1\_ui->Location = System::Drawing::Point(8, 20);

this->listing\_contractor1\_ui->Margin = System::Windows::Forms::Padding(4);

this->listing\_contractor1\_ui->Name = L"listing\_contractor1\_ui";

this->listing\_contractor1\_ui->Padding = System::Windows::Forms::Padding(4);

this->listing\_contractor1\_ui->Size = System::Drawing::Size(254, 147);

this->listing\_contractor1\_ui->TabIndex = 8;

this->listing\_contractor1\_ui->TabStop = false;

//

// listing\_contractor1\_button\_info

//

this->listing\_contractor1\_button\_info->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(187)),

static\_cast<System::Int32>(static\_cast<System::Byte>(191)), static\_cast<System::Int32>(static\_cast<System::Byte>(202)));

this->listing\_contractor1\_button\_info->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"listing\_contractor1\_button\_info.BackgroundImage")));

this->listing\_contractor1\_button\_info->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->listing\_contractor1\_button\_info->Location = System::Drawing::Point(5, 35);

this->listing\_contractor1\_button\_info->Margin = System::Windows::Forms::Padding(4);

this->listing\_contractor1\_button\_info->Name = L"listing\_contractor1\_button\_info";

this->listing\_contractor1\_button\_info->Size = System::Drawing::Size(30, 30);

this->listing\_contractor1\_button\_info->TabIndex = 9;

this->listing\_contractor1\_button\_info->UseVisualStyleBackColor = false;

this->listing\_contractor1\_button\_info->Click += gcnew System::EventHandler(this, &Main\_Menu::listing\_show\_author\_info);

//

// listing\_contractor1\_picker\_hrs

//

this->listing\_contractor1\_picker\_hrs->Location = System::Drawing::Point(114, 84);

this->listing\_contractor1\_picker\_hrs->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 8, 0, 0, 0 });

this->listing\_contractor1\_picker\_hrs->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0, 0, 0 });

this->listing\_contractor1\_picker\_hrs->Name = L"listing\_contractor1\_picker\_hrs";

this->listing\_contractor1\_picker\_hrs->Size = System::Drawing::Size(40, 28);

this->listing\_contractor1\_picker\_hrs->TabIndex = 9;

this->listing\_contractor1\_picker\_hrs->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0, 0, 0 });

this->listing\_contractor1\_picker\_hrs->ValueChanged += gcnew System::EventHandler(this, &Main\_Menu::feed\_contracts\_hrs\_adjusted);

//

// listing\_contractor1\_total

//

this->listing\_contractor1\_total->AutoSize = true;

this->listing\_contractor1\_total->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_contractor1\_total->Location = System::Drawing::Point(123, 117);

this->listing\_contractor1\_total->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor1\_total->Name = L"listing\_contractor1\_total";

this->listing\_contractor1\_total->Size = System::Drawing::Size(82, 20);

this->listing\_contractor1\_total->TabIndex = 7;

this->listing\_contractor1\_total->Text = L"Итого, ₽: ";

//

// listing\_contractor1\_button\_hire

//

this->listing\_contractor1\_button\_hire->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_contractor1\_button\_hire->Location = System::Drawing::Point(5, 109);

this->listing\_contractor1\_button\_hire->Margin = System::Windows::Forms::Padding(4);

this->listing\_contractor1\_button\_hire->Name = L"listing\_contractor1\_button\_hire";

this->listing\_contractor1\_button\_hire->Size = System::Drawing::Size(119, 34);

this->listing\_contractor1\_button\_hire->TabIndex = 7;

this->listing\_contractor1\_button\_hire->Text = L"Нанять!";

this->listing\_contractor1\_button\_hire->UseVisualStyleBackColor = true;

this->listing\_contractor1\_button\_hire->Click += gcnew System::EventHandler(this, &Main\_Menu::accept\_listing);

//

// listing\_contractor1\_label\_hrs

//

this->listing\_contractor1\_label\_hrs->AutoSize = true;

this->listing\_contractor1\_label\_hrs->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_contractor1\_label\_hrs->Location = System::Drawing::Point(5, 86);

this->listing\_contractor1\_label\_hrs->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor1\_label\_hrs->Name = L"listing\_contractor1\_label\_hrs";

this->listing\_contractor1\_label\_hrs->Size = System::Drawing::Size(109, 20);

this->listing\_contractor1\_label\_hrs->TabIndex = 5;

this->listing\_contractor1\_label\_hrs->Text = L"всего часов:";

//

// listing\_contractor1\_author

//

this->listing\_contractor1\_author->AutoSize = true;

this->listing\_contractor1\_author->Location = System::Drawing::Point(35, 40);

this->listing\_contractor1\_author->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor1\_author->Name = L"listing\_contractor1\_author";

this->listing\_contractor1\_author->Size = System::Drawing::Size(149, 20);

this->listing\_contractor1\_author->TabIndex = 2;

this->listing\_contractor1\_author->Text = L"имя\_подрядчика";

//

// listing\_contractor1\_label\_per\_hr

//

this->listing\_contractor1\_label\_per\_hr->AutoSize = true;

this->listing\_contractor1\_label\_per\_hr->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->listing\_contractor1\_label\_per\_hr->Location = System::Drawing::Point(5, 63);

this->listing\_contractor1\_label\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor1\_label\_per\_hr->Name = L"listing\_contractor1\_label\_per\_hr";

this->listing\_contractor1\_label\_per\_hr->Size = System::Drawing::Size(80, 20);

this->listing\_contractor1\_label\_per\_hr->TabIndex = 3;

this->listing\_contractor1\_label\_per\_hr->Text = L"за час, ₽:";

//

// listing\_contractor1\_per\_hr

//

this->listing\_contractor1\_per\_hr->AutoSize = true;

this->listing\_contractor1\_per\_hr->Location = System::Drawing::Point(89, 64);

this->listing\_contractor1\_per\_hr->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor1\_per\_hr->Name = L"listing\_contractor1\_per\_hr";

this->listing\_contractor1\_per\_hr->Size = System::Drawing::Size(117, 20);

this->listing\_contractor1\_per\_hr->TabIndex = 4;

this->listing\_contractor1\_per\_hr->Text = L"плата\_за\_час";

//

// listing\_contractor1\_name

//

this->listing\_contractor1\_name->AutoSize = true;

this->listing\_contractor1\_name->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->listing\_contractor1\_name->Location = System::Drawing::Point(4, 11);

this->listing\_contractor1\_name->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->listing\_contractor1\_name->Name = L"listing\_contractor1\_name";

this->listing\_contractor1\_name->Size = System::Drawing::Size(186, 24);

this->listing\_contractor1\_name->TabIndex = 0;

this->listing\_contractor1\_name->Text = L"название\_подряда";

//

// label\_feed

//

this->label\_feed->AutoSize = true;

this->label\_feed->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(232)), static\_cast<System::Int32>(static\_cast<System::Byte>(232)),

static\_cast<System::Int32>(static\_cast<System::Byte>(232)));

this->label\_feed->Font = (gcnew System::Drawing::Font(L"Roboto", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_feed->Location = System::Drawing::Point(7, 7);

this->label\_feed->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_feed->Name = L"label\_feed";

this->label\_feed->Size = System::Drawing::Size(214, 28);

this->label\_feed->TabIndex = 0;

this->label\_feed->Text = L"Лента объявлений";

//

// bg\_feed

//

this->bg\_feed->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"bg\_feed.BackgroundImage")));

this->bg\_feed->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->bg\_feed->Location = System::Drawing::Point(0, -7);

this->bg\_feed->Margin = System::Windows::Forms::Padding(4);

this->bg\_feed->Name = L"bg\_feed";

this->bg\_feed->Size = System::Drawing::Size(560, 169);

this->bg\_feed->TabIndex = 1;

this->bg\_feed->TabStop = false;

//

// update\_timer

//

this->update\_timer->Interval = 500;

this->update\_timer->Tick += gcnew System::EventHandler(this, &Main\_Menu::update\_feed);

//

// menu\_search

//

this->menu\_search->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->menu\_search->Controls->Add(this->button\_filter);

this->menu\_search->Controls->Add(this->label\_in\_future\_updates);

this->menu\_search->Controls->Add(this->label\_search\_will\_be);

this->menu\_search->Controls->Add(this->button\_search);

this->menu\_search->Controls->Add(this->input\_search);

this->menu\_search->Controls->Add(this->label\_search);

this->menu\_search->Controls->Add(this->bg\_search);

this->menu\_search->Location = System::Drawing::Point(10, 11);

this->menu\_search->Margin = System::Windows::Forms::Padding(4);

this->menu\_search->Name = L"menu\_search";

this->menu\_search->Size = System::Drawing::Size(560, 502);

this->menu\_search->TabIndex = 7;

//

// button\_filter

//

this->button\_filter->BackColor = System::Drawing::SystemColors::ControlLight;

this->button\_filter->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"button\_filter.BackgroundImage")));

this->button\_filter->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->button\_filter->Enabled = false;

this->button\_filter->Location = System::Drawing::Point(38, 168);

this->button\_filter->Margin = System::Windows::Forms::Padding(4);

this->button\_filter->Name = L"button\_filter";

this->button\_filter->Size = System::Drawing::Size(35, 37);

this->button\_filter->TabIndex = 6;

this->button\_filter->UseVisualStyleBackColor = false;

//

// label\_in\_future\_updates

//

this->label\_in\_future\_updates->AutoSize = true;

this->label\_in\_future\_updates->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_in\_future\_updates->Location = System::Drawing::Point(75, 248);

this->label\_in\_future\_updates->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_in\_future\_updates->Name = L"label\_in\_future\_updates";

this->label\_in\_future\_updates->Size = System::Drawing::Size(411, 20);

this->label\_in\_future\_updates->TabIndex = 4;

this->label\_in\_future\_updates->Text = L"...в будущих обновлениях. Спасибо, что вы с нами!\r\n";

//

// label\_search\_will\_be

//

this->label\_search\_will\_be->AutoSize = true;

this->label\_search\_will\_be->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_search\_will\_be->Location = System::Drawing::Point(213, 216);

this->label\_search\_will\_be->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_search\_will\_be->Name = L"label\_search\_will\_be";

this->label\_search\_will\_be->Size = System::Drawing::Size(149, 24);

this->label\_search\_will\_be->TabIndex = 3;

this->label\_search\_will\_be->Text = L"Поиску - быть!";

//

// button\_search

//

this->button\_search->BackColor = System::Drawing::SystemColors::ControlLight;

this->button\_search->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"button\_search.BackgroundImage")));

this->button\_search->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->button\_search->Enabled = false;

this->button\_search->Location = System::Drawing::Point(4, 168);

this->button\_search->Margin = System::Windows::Forms::Padding(4);

this->button\_search->Name = L"button\_search";

this->button\_search->Size = System::Drawing::Size(35, 37);

this->button\_search->TabIndex = 2;

this->button\_search->UseVisualStyleBackColor = false;

//

// input\_search

//

this->input\_search->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_search->Location = System::Drawing::Point(79, 173);

this->input\_search->Margin = System::Windows::Forms::Padding(4);

this->input\_search->Name = L"input\_search";

this->input\_search->Size = System::Drawing::Size(472, 30);

this->input\_search->TabIndex = 1;

//

// label\_search

//

this->label\_search->AutoSize = true;

this->label\_search->Font = (gcnew System::Drawing::Font(L"Roboto", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_search->Location = System::Drawing::Point(7, 7);

this->label\_search->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label\_search->Name = L"label\_search";

this->label\_search->Size = System::Drawing::Size(202, 28);

this->label\_search->TabIndex = 0;

this->label\_search->Text = L"Поиск и фильтры";

//

// bg\_search

//

this->bg\_search->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"bg\_search.BackgroundImage")));

this->bg\_search->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->bg\_search->Location = System::Drawing::Point(0, -7);

this->bg\_search->Margin = System::Windows::Forms::Padding(4);

this->bg\_search->Name = L"bg\_search";

this->bg\_search->Size = System::Drawing::Size(560, 169);

this->bg\_search->TabIndex = 5;

this->bg\_search->TabStop = false;

//

// Main\_Menu

//

this->AutoScaleDimensions = System::Drawing::SizeF(7, 15);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::SystemColors::Control;

this->ClientSize = System::Drawing::Size(684, 517);

this->Controls->Add(this->sidebar);

this->Controls->Add(this->menu\_account);

this->Controls->Add(this->menu\_search);

this->Controls->Add(this->menu\_feed);

this->DoubleBuffered = true;

this->Font = (gcnew System::Drawing::Font(L"Roboto", 8, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->Margin = System::Windows::Forms::Padding(4);

this->MaximumSize = System::Drawing::Size(702, 564);

this->MinimumSize = System::Drawing::Size(702, 564);

this->Name = L"Main\_Menu";

this->Text = L"Agora";

this->sidebar->ResumeLayout(false);

this->menu\_account->ResumeLayout(false);

this->menu\_account->PerformLayout();

this->group\_active\_contracts->ResumeLayout(false);

this->active\_listing2\_ui->ResumeLayout(false);

this->active\_listing2\_ui->PerformLayout();

this->active\_listing1\_ui->ResumeLayout(false);

this->active\_listing1\_ui->PerformLayout();

this->group\_account\_data->ResumeLayout(false);

this->group\_account\_data->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->bg\_my\_account))->EndInit();

this->menu\_feed->ResumeLayout(false);

this->menu\_feed->PerformLayout();

this->group\_customers->ResumeLayout(false);

this->listing\_customer2\_ui->ResumeLayout(false);

this->listing\_customer2\_ui->PerformLayout();

this->listing\_customer1\_ui->ResumeLayout(false);

this->listing\_customer1\_ui->PerformLayout();

this->group\_contractors->ResumeLayout(false);

this->listing\_contractor2\_ui->ResumeLayout(false);

this->listing\_contractor2\_ui->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->listing\_contractor2\_picker\_hrs))->EndInit();

this->listing\_contractor1\_ui->ResumeLayout(false);

this->listing\_contractor1\_ui->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->listing\_contractor1\_picker\_hrs))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->bg\_feed))->EndInit();

this->menu\_search->ResumeLayout(false);

this->menu\_search->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->bg\_search))->EndInit();

this->ResumeLayout(false);

}

#pragma endregion

private:

#pragma region ======= Feed =======

void sidebar\_pick\_feed(System::Object^ sender, System::EventArgs^ e)

{

this->Text = "Agora: Лента";

menu\_feed->BringToFront();

}

void accept\_listing(System::Object^ sender, System::EventArgs^ e)

{

Button^ source = safe\_cast<Button^>(sender);

Listing\_Interface^ respectable\_listing\_ui;

bool accepted = randint\_percent(80);

if (contractor1->has(source))

{

respectable\_listing\_ui = contractor1;

contractor1->listing->customer = user;

if (accepted)

show\_info (L"Подрядчик согласился!");

else

show\_error(L"Подрядчик отказался!", L"Отказ");

}

else if (contractor2->has(source))

{

respectable\_listing\_ui = contractor2;

contractor2->listing->customer = user;

if (accepted)

show\_info (L"Подрядчик согласился!");

else

show\_error(L"Подрядчик отказался!", L"Отказ");

}

else if (customer1->has(source))

{

respectable\_listing\_ui = customer1;

customer1->listing->contractor = user;

if (accepted)

show\_info (L"Заказчик согласился!");

else

show\_error(L"Заказчик отказался!", L"Отказ");

}

else if (customer2->has(source))

{

respectable\_listing\_ui = customer2;

customer2->listing->contractor = user;

if (accepted)

show\_info (L"Заказчик согласился!");

else

show\_error(L"Заказчик отказался!", L"Отказ");

}

else return;

if (accepted)

{

active\_listings.push\_back(\*respectable\_listing\_ui->listing);

save(user, USER\_SAVEFILE\_NAME, active\_listings);

fill\_account\_listing\_interface();

}

respectable\_listing\_ui->ui\_group->Enabled = false;

}

void feed\_contracts\_hrs\_adjusted(System::Object^ sender, System::EventArgs^ e)

{

NumericUpDown^ source = safe\_cast<NumericUpDown^>(sender);

Listing\_Interface^ respectable\_listing\_ui;

if (contractor1->has(source))

respectable\_listing\_ui = contractor1;

else if (contractor2->has(source))

respectable\_listing\_ui = contractor2;

else return;

respectable\_listing\_ui->calculate\_and\_display\_total(Convert::ToUInt32(source->Value));

}

void fill\_feed\_menu()

{

listing\_contractor1 = get\_random\_listing(CONTRACTOR\_LISTING);

contractor1->set\_listing(&listing\_contractor1);

listing\_contractor2 = get\_random\_listing(CONTRACTOR\_LISTING);

contractor2->set\_listing(&listing\_contractor2);

listing\_customer1 = get\_random\_listing(CUSTOMER\_LISTING);

customer1->set\_listing(&listing\_customer1);

listing\_customer2 = get\_random\_listing(CUSTOMER\_LISTING);

customer2->set\_listing(&listing\_customer2);

}

void listing\_show\_author\_info(System::Object^ sender, System::EventArgs^ e)

{

Button^ source = safe\_cast<Button^>(sender);

if (contractor1->has(source))

contractor1->show\_info();

else if (contractor2->has(source))

contractor2->show\_info();

else if (customer1->has(source))

customer1->show\_info();

else if (customer2->has(source))

customer2->show\_info();

}

void update\_feed(System::Object^ \_sender, System::EventArgs^ e)

{

/// Think I'm a genius. This function is used both for start and end of the 'update'. Elegant!

button\_feed\_update->Enabled = ! button\_feed\_update->Enabled;

group\_contractors->Enabled = ! group\_contractors->Enabled;

group\_customers->Enabled = ! group\_customers->Enabled;

update\_timer->Enabled = ! update\_timer->Enabled;

/// countdown finished

if (! update\_timer->Enabled)

fill\_feed\_menu();

}

#pragma endregion

#pragma region ======= Account =======

void sidebar\_pick\_account(System::Object^ sender, System::EventArgs^ e)

{

this->Text = "Agora: Мой аккаунт";

menu\_account->BringToFront();

Point label\_pos(11, 113);

label\_no\_active\_contracts->Location = label\_pos;

label\_no\_active\_contracts->Visible = active\_listings.empty();

fill\_account\_listing\_interface();

}

void edit\_account\_data(System::Object^ sender, System::EventArgs^ e)

{

Registration^ registration = gcnew Registration(user);

registration->ShowDialog();

user = load(USER\_SAVEFILE\_NAME);

if (user == nullptr)

{

show\_error(L"Ошибка при чтении данных.");

this->Close();

return;

}

fill\_account\_menu();

}

void fill\_account\_menu()

{

output\_account\_name->Text = to\_dotnet\_string(user->name->get\_normal());

if (typeid(\*user) == typeid(Individual))

{

label\_account\_website-> Enabled = false;

output\_account\_website->Enabled = false;

label\_account\_age->Text = "Возраст:";

output\_account\_age->Text = to\_dotnet\_string(std::to\_wstring(user->birth\_date.years\_since()) + L" лет");

}

else if (typeid(\*user) == typeid(Company))

{

output\_account\_website->Text = to\_dotnet\_string(user->website);

label\_account\_website-> Enabled = true;

output\_account\_website->Enabled = true;

label\_account\_age->Text = "На рынке с:";

output\_account\_age->Text = to\_dotnet\_string(user->birth\_date.month\_yyyy\_genitive());

}

output\_account\_phone\_number->Text = to\_dotnet\_string(user->phone\_number.as\_text());

output\_account\_email->Text = to\_dotnet\_string(user->email);

if (output\_account\_email->Text == "")

output\_account\_email->Text = "-";

output\_account\_extra->Text = to\_dotnet\_string(user->extra\_contacts);

if (output\_account\_extra->Text == "")

output\_account\_extra->Text = "-";

}

void active\_listing\_show\_author\_info(System::Object^ sender, System::EventArgs^ e)

{

Button^ source = safe\_cast<Button^>(sender);

if (active\_listing1->has(source))

active\_listing1->show\_info();

else if (active\_listing2->has(source))

active\_listing2->show\_info();

}

void finish\_active\_listing(System::Object^ sender, System::EventArgs^ e)

{

Button^ source = safe\_cast<Button^>(sender);

if (active\_listing1->has(source))

{

active\_listing1\_ui->Enabled = false;

active\_listings.erase(find(active\_listings.begin(), active\_listings.end(), \*active\_listing1->listing));

}

else if (active\_listing2->has(source))

{

active\_listing2\_ui->Enabled = false;

active\_listings.erase(find(active\_listings.begin(), active\_listings.end(), \*active\_listing2->listing));

}

save(user, USER\_SAVEFILE\_NAME, active\_listings);

}

#pragma endregion

#pragma region ======= Misc. menus =======

void sidebar\_pick\_info(System::Object^ sender, System::EventArgs^ e)

{

show\_info(PROGRAM\_INFO, L"О программе");

}

void sidebar\_pick\_search(System::Object^ sender, System::EventArgs^ e)

{

this->Text = "Agora: Поиск объявлений и аккаунтов";

menu\_search->BringToFront();

}

#pragma endregion

};

}

#endif

#### Листинг кода registration.hpp

#pragma once

#ifndef REGISTRATION\_HPP

#define REGISTRATION\_HPP

#include "Classes/listing.hpp"

#include "Classes/user.hpp"

#include "../../save\_load.hpp"

#include <string>

#include <vector>

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

namespace Agora

{

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

{

public:

User\* user;

Registration(User\* \_user)

: user(\_user)

{

InitializeComponent();

pick\_as\_individual(nullptr, nullptr);

System::Drawing::Font^ font = gcnew System::Drawing::Font("Roboto", 7);

this->Font = font;

if (user == nullptr) // fresh user

{

set\_greeting\_time\_based();

return;

}

/// = Re-registration =

this->Text = "Редактирование данных аккаунта";

label\_registration->Text = "Редактирование";

label\_kalimera->Text = "Изменение данных.";

label\_no\_account->Text = "Текущие данные вашего аккаунта введены.";

std::vector<std::wstring> name = user->name->as\_vector();

std::vector<std::wstring> birth\_date = user->birth\_date.as\_vector();

if (typeid(\*user) == typeid(Individual))

{

button\_individual->Select();

pick\_as\_individual(nullptr, nullptr);

this->input\_surname->Text = to\_dotnet\_string(name[0]);

this->input\_name->Text = to\_dotnet\_string(name[1]);

this->input\_patronym->Text = to\_dotnet\_string(name[2]);

this->input\_ind\_day->Text = to\_dotnet\_string(birth\_date[0]);

this->input\_ind\_month->SelectedIndex = \_wtoi(birth\_date[1].c\_str()) - 1;

this->input\_ind\_year->Text = to\_dotnet\_string(birth\_date[2]);

}

else if (typeid(\*user) == typeid(Company))

{

button\_company->Select();

pick\_as\_company(nullptr, nullptr);

this->input\_legal\_form->SelectedItem = to\_dotnet\_string(name[0]);

this->input\_company\_name->Text = to\_dotnet\_string(name[1]);

this->input\_com\_month->SelectedIndex = \_wtoi(birth\_date[1].c\_str()) - 1;

this->input\_com\_year->Text = to\_dotnet\_string(birth\_date[2]);

this->input\_website->Text = to\_dotnet\_string(user->website);

}

else

{

show\_error(L"Type error: couldn't resolve user's account type");

this->Close();

}

/// = Common between both account types =

std::vector<std::wstring> phone = separate(user->phone\_number.as\_text());

this->input\_country\_code->Text = to\_dotnet\_string(phone[0]);

this->input\_carrier\_code->Text = to\_dotnet\_string(phone[1]);

this->input\_phone\_number\_body->Text = to\_dotnet\_string(phone[2]);

this->input\_email->Text = to\_dotnet\_string(user->email);

this->input\_extra->Text = to\_dotnet\_string(user->extra\_contacts);

}

private:

void center\_x(System::Object^ sender, System::EventArgs^ e)

{

Control^ target = safe\_cast<Control^>(sender);

/// Don't y'all love variables that don't state they're readonly?

System::Drawing::Size size = TextRenderer::MeasureText(target->Text, target->Font);

target->Size = size;

System::Drawing::Point location((this->Width - size.Width) / 2, target->Location.Y);

target->Location = location;

}

void set\_greeting\_time\_based()

{

int time = get\_time();

if (time < 40000)

label\_kalimera->Text += "ый вечер!";

else if (time < 120000)

label\_kalimera->Text += "ое утро!";

else if (time < 170000)

label\_kalimera->Text += "ый день!";

else

label\_kalimera->Text += "ый вечер!";

}

#pragma region ========== WinForms code ==========

protected:

~Registration()

{

if (components)

delete components;

}

private:

System::Windows::Forms::Label^ label\_registration;

System::Windows::Forms::Label^ label\_kalimera;

System::Windows::Forms::Label^ label\_no\_account;

System::Windows::Forms::RadioButton^ button\_individual;

System::Windows::Forms::RadioButton^ button\_company;

System::Windows::Forms::GroupBox^ registration\_individual;

System::Windows::Forms::Label^ label\_name;

System::Windows::Forms::Label^ label\_patronym;

System::Windows::Forms::TextBox^ input\_patronym;

System::Windows::Forms::Label^ label\_surname;

System::Windows::Forms::TextBox^ input\_surname;

System::Windows::Forms::Label^ label\_if\_you\_have\_patronym;

System::Windows::Forms::TextBox^ input\_country\_code;

System::Windows::Forms::TextBox^ input\_carrier\_code;

System::Windows::Forms::Label^ label\_phone;

System::Windows::Forms::TextBox^ input\_phone\_number\_body;

System::Windows::Forms::GroupBox^ contacts;

System::Windows::Forms::Label^ label\_email;

System::Windows::Forms::TextBox^ input\_email;

System::Windows::Forms::Label^ label\_if\_you\_have\_email;

System::Windows::Forms::Label^ label\_if\_you\_have\_extra;

System::Windows::Forms::TextBox^ input\_extra;

System::Windows::Forms::Label^ label\_extra;

System::Windows::Forms::GroupBox^ registration\_company;

System::Windows::Forms::Label^ label\_company\_name;

System::Windows::Forms::TextBox^ input\_company\_name;

System::Windows::Forms::Button^ button\_register;

System::Windows::Forms::ComboBox^ input\_legal\_form;

System::Windows::Forms::Label^ label\_website;

System::Windows::Forms::Label^ label\_est;

System::Windows::Forms::TextBox^ input\_website;

System::Windows::Forms::Label^ label\_month;

System::Windows::Forms::Label^ label\_year;

System::Windows::Forms::TextBox^ input\_name;

System::Windows::Forms::TextBox^ input\_ind\_day;

System::Windows::Forms::Label^ label\_birthdate;

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

System::Windows::Forms::TextBox^ input\_ind\_year;

System::Windows::Forms::Label^ label\_y;

System::Windows::Forms::TextBox^ input\_com\_year;

System::Windows::Forms::ComboBox^ input\_com\_month;

System::Windows::Forms::ComboBox^ input\_ind\_month;

System::ComponentModel::Container ^components;

void InitializeComponent(void)

{

this->label\_registration = (gcnew System::Windows::Forms::Label());

this->label\_kalimera = (gcnew System::Windows::Forms::Label());

this->label\_no\_account = (gcnew System::Windows::Forms::Label());

this->button\_individual = (gcnew System::Windows::Forms::RadioButton());

this->button\_company = (gcnew System::Windows::Forms::RadioButton());

this->registration\_individual = (gcnew System::Windows::Forms::GroupBox());

this->input\_ind\_month = (gcnew System::Windows::Forms::ComboBox());

this->input\_ind\_day = (gcnew System::Windows::Forms::TextBox());

this->label\_birthdate = (gcnew System::Windows::Forms::Label());

this->label1 = (gcnew System::Windows::Forms::Label());

this->input\_ind\_year = (gcnew System::Windows::Forms::TextBox());

this->input\_name = (gcnew System::Windows::Forms::TextBox());

this->label\_if\_you\_have\_patronym = (gcnew System::Windows::Forms::Label());

this->label\_patronym = (gcnew System::Windows::Forms::Label());

this->input\_patronym = (gcnew System::Windows::Forms::TextBox());

this->label\_surname = (gcnew System::Windows::Forms::Label());

this->input\_surname = (gcnew System::Windows::Forms::TextBox());

this->label\_name = (gcnew System::Windows::Forms::Label());

this->input\_phone\_number\_body = (gcnew System::Windows::Forms::TextBox());

this->input\_country\_code = (gcnew System::Windows::Forms::TextBox());

this->input\_carrier\_code = (gcnew System::Windows::Forms::TextBox());

this->label\_phone = (gcnew System::Windows::Forms::Label());

this->contacts = (gcnew System::Windows::Forms::GroupBox());

this->label\_if\_you\_have\_extra = (gcnew System::Windows::Forms::Label());

this->input\_extra = (gcnew System::Windows::Forms::TextBox());

this->label\_extra = (gcnew System::Windows::Forms::Label());

this->label\_if\_you\_have\_email = (gcnew System::Windows::Forms::Label());

this->input\_email = (gcnew System::Windows::Forms::TextBox());

this->label\_email = (gcnew System::Windows::Forms::Label());

this->registration\_company = (gcnew System::Windows::Forms::GroupBox());

this->label\_year = (gcnew System::Windows::Forms::Label());

this->label\_month = (gcnew System::Windows::Forms::Label());

this->input\_website = (gcnew System::Windows::Forms::TextBox());

this->label\_y = (gcnew System::Windows::Forms::Label());

this->input\_com\_year = (gcnew System::Windows::Forms::TextBox());

this->input\_com\_month = (gcnew System::Windows::Forms::ComboBox());

this->label\_website = (gcnew System::Windows::Forms::Label());

this->label\_est = (gcnew System::Windows::Forms::Label());

this->input\_legal\_form = (gcnew System::Windows::Forms::ComboBox());

this->label\_company\_name = (gcnew System::Windows::Forms::Label());

this->input\_company\_name = (gcnew System::Windows::Forms::TextBox());

this->button\_register = (gcnew System::Windows::Forms::Button());

this->registration\_individual->SuspendLayout();

this->contacts->SuspendLayout();

this->registration\_company->SuspendLayout();

this->SuspendLayout();

//

// label\_registration

//

this->label\_registration->AutoSize = true;

this->label\_registration->Font = (gcnew System::Drawing::Font(L"Roboto", 16.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_registration->Location = System::Drawing::Point(182, 8);

this->label\_registration->Name = L"label\_registration";

this->label\_registration->Size = System::Drawing::Size(185, 34);

this->label\_registration->TabIndex = 1000;

this->label\_registration->Text = L"Регистрация";

this->label\_registration->TextChanged += gcnew System::EventHandler(this, &Registration::center\_x);

//

// label\_kalimera

//

this->label\_kalimera->AutoSize = true;

this->label\_kalimera->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_kalimera->Location = System::Drawing::Point(164, 48);

this->label\_kalimera->Name = L"label\_kalimera";

this->label\_kalimera->Size = System::Drawing::Size(58, 24);

this->label\_kalimera->TabIndex = 1000;

this->label\_kalimera->Text = L"Добр";

this->label\_kalimera->TextChanged += gcnew System::EventHandler(this, &Registration::center\_x);

//

// label\_no\_account

//

this->label\_no\_account->AutoSize = true;

this->label\_no\_account->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_no\_account->Location = System::Drawing::Point(50, 79);

this->label\_no\_account->Name = L"label\_no\_account";

this->label\_no\_account->Size = System::Drawing::Size(461, 24);

this->label\_no\_account->TabIndex = 1000;

this->label\_no\_account->Text = L"Ваш аккаунт не был обнаружен в системе. Вы...";

this->label\_no\_account->TextChanged += gcnew System::EventHandler(this, &Registration::center\_x);

//

// button\_individual

//

this->button\_individual->AutoSize = true;

this->button\_individual->Checked = true;

this->button\_individual->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button\_individual->Location = System::Drawing::Point(116, 108);

this->button\_individual->Name = L"button\_individual";

this->button\_individual->Size = System::Drawing::Size(125, 28);

this->button\_individual->TabIndex = 7;

this->button\_individual->TabStop = true;

this->button\_individual->Text = L"Физ. лицо";

this->button\_individual->UseVisualStyleBackColor = true;

this->button\_individual->CheckedChanged += gcnew System::EventHandler(this, &Registration::pick\_as\_individual);

//

// button\_company

//

this->button\_company->AutoSize = true;

this->button\_company->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button\_company->Location = System::Drawing::Point(298, 108);

this->button\_company->Name = L"button\_company";

this->button\_company->Size = System::Drawing::Size(117, 28);

this->button\_company->TabIndex = 8;

this->button\_company->Text = L"Юр. лицо";

this->button\_company->UseVisualStyleBackColor = true;

this->button\_company->CheckedChanged += gcnew System::EventHandler(this, &Registration::pick\_as\_company);

//

// registration\_individual

//

this->registration\_individual->Controls->Add(this->input\_ind\_month);

this->registration\_individual->Controls->Add(this->input\_ind\_day);

this->registration\_individual->Controls->Add(this->label\_birthdate);

this->registration\_individual->Controls->Add(this->label1);

this->registration\_individual->Controls->Add(this->input\_ind\_year);

this->registration\_individual->Controls->Add(this->input\_name);

this->registration\_individual->Controls->Add(this->label\_if\_you\_have\_patronym);

this->registration\_individual->Controls->Add(this->label\_patronym);

this->registration\_individual->Controls->Add(this->input\_patronym);

this->registration\_individual->Controls->Add(this->label\_surname);

this->registration\_individual->Controls->Add(this->input\_surname);

this->registration\_individual->Controls->Add(this->label\_name);

this->registration\_individual->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->registration\_individual->Location = System::Drawing::Point(39, 141);

this->registration\_individual->Name = L"registration\_individual";

this->registration\_individual->Size = System::Drawing::Size(410, 172);

this->registration\_individual->TabIndex = 0;

this->registration\_individual->TabStop = false;

this->registration\_individual->Text = L"ФИО";

//

// input\_ind\_month

//

this->input\_ind\_month->AutoCompleteMode = System::Windows::Forms::AutoCompleteMode::Append;

this->input\_ind\_month->AutoCompleteSource = System::Windows::Forms::AutoCompleteSource::ListItems;

this->input\_ind\_month->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->input\_ind\_month->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_ind\_month->FormattingEnabled = true;

this->input\_ind\_month->Items->AddRange(gcnew cli::array< System::Object^ >(12) {

L"января", L"февраля", L"марта", L"апреля",

L"мая", L"июня", L"июля", L"августа", L"сентября", L"октября", L"ноября", L"декабря"

});

this->input\_ind\_month->Location = System::Drawing::Point(213, 137);

this->input\_ind\_month->MaxLength = 4;

this->input\_ind\_month->Name = L"input\_ind\_month";

this->input\_ind\_month->Size = System::Drawing::Size(85, 28);

this->input\_ind\_month->TabIndex = 1012;

this->input\_ind\_month->SelectedValueChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_individual\_registration\_check);

//

// input\_ind\_day

//

this->input\_ind\_day->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_ind\_day->Location = System::Drawing::Point(165, 136);

this->input\_ind\_day->MaxLength = 2;

this->input\_ind\_day->Name = L"input\_ind\_day";

this->input\_ind\_day->Size = System::Drawing::Size(42, 30);

this->input\_ind\_day->TabIndex = 3;

this->input\_ind\_day->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_ind\_day->TextChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_individual\_registration\_check);

this->input\_ind\_day->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::only\_digits);

//

// label\_birthdate

//

this->label\_birthdate->AutoSize = true;

this->label\_birthdate->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_birthdate->Location = System::Drawing::Point(14, 138);

this->label\_birthdate->Name = L"label\_birthdate";

this->label\_birthdate->Size = System::Drawing::Size(167, 25);

this->label\_birthdate->TabIndex = 1010;

this->label\_birthdate->Text = L"Дата рождения:";

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(373, 138);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(25, 25);

this->label1->TabIndex = 1009;

this->label1->Text = L"г.";

//

// input\_ind\_year

//

this->input\_ind\_year->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_ind\_year->Location = System::Drawing::Point(303, 136);

this->input\_ind\_year->MaxLength = 4;

this->input\_ind\_year->Name = L"input\_ind\_year";

this->input\_ind\_year->Size = System::Drawing::Size(65, 30);

this->input\_ind\_year->TabIndex = 4;

this->input\_ind\_year->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_ind\_year->TextChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_individual\_registration\_check);

this->input\_ind\_year->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::only\_digits);

//

// input\_name

//

this->input\_name->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_name->Location = System::Drawing::Point(116, 52);

this->input\_name->MaxLength = 40;

this->input\_name->Name = L"input\_name";

this->input\_name->Size = System::Drawing::Size(280, 30);

this->input\_name->TabIndex = 1001;

this->input\_name->TabStop = false;

this->input\_name->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_name->TextChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_individual\_registration\_check);

this->input\_name->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::only\_letters);

//

// label\_if\_you\_have\_patronym

//

this->label\_if\_you\_have\_patronym->AutoSize = true;

this->label\_if\_you\_have\_patronym->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->label\_if\_you\_have\_patronym->Location = System::Drawing::Point(210, 118);

this->label\_if\_you\_have\_patronym->Name = L"label\_if\_you\_have\_patronym";

this->label\_if\_you\_have\_patronym->Size = System::Drawing::Size(102, 17);

this->label\_if\_you\_have\_patronym->TabIndex = 1000;

this->label\_if\_you\_have\_patronym->Text = L"(при наличии)";

//

// label\_patronym

//

this->label\_patronym->AutoSize = true;

this->label\_patronym->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_patronym->Location = System::Drawing::Point(14, 90);

this->label\_patronym->Name = L"label\_patronym";

this->label\_patronym->Size = System::Drawing::Size(110, 25);

this->label\_patronym->TabIndex = 1000;

this->label\_patronym->Text = L"Отчество:";

//

// input\_patronym

//

this->input\_patronym->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_patronym->Location = System::Drawing::Point(116, 87);

this->input\_patronym->MaxLength = 40;

this->input\_patronym->Name = L"input\_patronym";

this->input\_patronym->Size = System::Drawing::Size(280, 30);

this->input\_patronym->TabIndex = 2;

this->input\_patronym->TabStop = false;

this->input\_patronym->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_patronym->TextChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_individual\_registration\_check);

this->input\_patronym->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::only\_letters);

//

// label\_surname

//

this->label\_surname->AutoSize = true;

this->label\_surname->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_surname->Location = System::Drawing::Point(14, 22);

this->label\_surname->Name = L"label\_surname";

this->label\_surname->Size = System::Drawing::Size(109, 25);

this->label\_surname->TabIndex = 1000;

this->label\_surname->Text = L"Фамилия:";

//

// input\_surname

//

this->input\_surname->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_surname->Location = System::Drawing::Point(116, 20);

this->input\_surname->MaxLength = 40;

this->input\_surname->Name = L"input\_surname";

this->input\_surname->Size = System::Drawing::Size(280, 30);

this->input\_surname->TabIndex = 0;

this->input\_surname->TabStop = false;

this->input\_surname->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_surname->TextChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_individual\_registration\_check);

this->input\_surname->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::only\_letters);

//

// label\_name

//

this->label\_name->AutoSize = true;

this->label\_name->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_name->Location = System::Drawing::Point(14, 56);

this->label\_name->Name = L"label\_name";

this->label\_name->Size = System::Drawing::Size(60, 25);

this->label\_name->TabIndex = 1000;

this->label\_name->Text = L"Имя:";

//

// input\_phone\_number\_body

//

this->input\_phone\_number\_body->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_phone\_number\_body->Location = System::Drawing::Point(303, 22);

this->input\_phone\_number\_body->MaxLength = 9;

this->input\_phone\_number\_body->Name = L"input\_phone\_number\_body";

this->input\_phone\_number\_body->Size = System::Drawing::Size(98, 32);

this->input\_phone\_number\_body->TabIndex = 4;

this->input\_phone\_number\_body->TabStop = false;

this->input\_phone\_number\_body->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_phone\_number\_body->TextChanged += gcnew System::EventHandler(this, &Registration::on\_phone\_number\_body\_input);

this->input\_phone\_number\_body->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::only\_digits);

this->input\_phone\_number\_body->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

//

// input\_country\_code

//

this->input\_country\_code->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_country\_code->Location = System::Drawing::Point(209, 22);

this->input\_country\_code->Name = L"input\_country\_code";

this->input\_country\_code->ReadOnly = true;

this->input\_country\_code->Size = System::Drawing::Size(34, 32);

this->input\_country\_code->TabIndex = 1000;

this->input\_country\_code->TabStop = false;

this->input\_country\_code->Text = L"+7";

//

// input\_carrier\_code

//

this->input\_carrier\_code->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_carrier\_code->Location = System::Drawing::Point(248, 22);

this->input\_carrier\_code->MaxLength = 5;

this->input\_carrier\_code->Name = L"input\_carrier\_code";

this->input\_carrier\_code->Size = System::Drawing::Size(52, 32);

this->input\_carrier\_code->TabIndex = 3;

this->input\_carrier\_code->TabStop = false;

this->input\_carrier\_code->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_carrier\_code->TextChanged += gcnew System::EventHandler(this, &Registration::on\_carrier\_code\_input);

this->input\_carrier\_code->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::only\_digits);

this->input\_carrier\_code->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

//

// label\_phone

//

this->label\_phone->AutoSize = true;

this->label\_phone->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_phone->Location = System::Drawing::Point(14, 26);

this->label\_phone->Name = L"label\_phone";

this->label\_phone->Size = System::Drawing::Size(174, 24);

this->label\_phone->TabIndex = 1000;

this->label\_phone->Text = L"Номер телефона:";

//

// contacts

//

this->contacts->Controls->Add(this->input\_phone\_number\_body);

this->contacts->Controls->Add(this->label\_if\_you\_have\_extra);

this->contacts->Controls->Add(this->input\_carrier\_code);

this->contacts->Controls->Add(this->input\_country\_code);

this->contacts->Controls->Add(this->input\_extra);

this->contacts->Controls->Add(this->label\_extra);

this->contacts->Controls->Add(this->label\_phone);

this->contacts->Controls->Add(this->label\_if\_you\_have\_email);

this->contacts->Controls->Add(this->input\_email);

this->contacts->Controls->Add(this->label\_email);

this->contacts->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->contacts->Location = System::Drawing::Point(39, 318);

this->contacts->Name = L"contacts";

this->contacts->Size = System::Drawing::Size(410, 142);

this->contacts->TabIndex = 3;

this->contacts->TabStop = false;

this->contacts->Text = L"Контакты";

//

// label\_if\_you\_have\_extra

//

this->label\_if\_you\_have\_extra->AutoSize = true;

this->label\_if\_you\_have\_extra->Font = (gcnew System::Drawing::Font(L"Roboto", 7.8F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_if\_you\_have\_extra->Location = System::Drawing::Point(16, 118);

this->label\_if\_you\_have\_extra->Name = L"label\_if\_you\_have\_extra";

this->label\_if\_you\_have\_extra->Size = System::Drawing::Size(96, 15);

this->label\_if\_you\_have\_extra->TabIndex = 1000;

this->label\_if\_you\_have\_extra->Text = L"(при наличии)";

//

// input\_extra

//

this->input\_extra->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_extra->Location = System::Drawing::Point(167, 91);

this->input\_extra->MaxLength = 50;

this->input\_extra->Name = L"input\_extra";

this->input\_extra->Size = System::Drawing::Size(233, 32);

this->input\_extra->TabIndex = 6;

this->input\_extra->TabStop = false;

this->input\_extra->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_extra->TextChanged += gcnew System::EventHandler(this, &Registration::common\_contacts\_sufficient\_input\_check);

this->input\_extra->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::keyboard\_focus\_switch\_check);

this->input\_extra->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

//

// label\_extra

//

this->label\_extra->AutoSize = true;

this->label\_extra->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_extra->Location = System::Drawing::Point(14, 95);

this->label\_extra->Name = L"label\_extra";

this->label\_extra->Size = System::Drawing::Size(166, 24);

this->label\_extra->TabIndex = 1000;

this->label\_extra->Text = L"Дополнительно:";

//

// label\_if\_you\_have\_email

//

this->label\_if\_you\_have\_email->AutoSize = true;

this->label\_if\_you\_have\_email->Font = (gcnew System::Drawing::Font(L"Roboto", 7.8F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_if\_you\_have\_email->Location = System::Drawing::Point(16, 79);

this->label\_if\_you\_have\_email->Name = L"label\_if\_you\_have\_email";

this->label\_if\_you\_have\_email->Size = System::Drawing::Size(96, 15);

this->label\_if\_you\_have\_email->TabIndex = 1000;

this->label\_if\_you\_have\_email->Text = L"(при наличии)";

//

// input\_email

//

this->input\_email->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_email->Location = System::Drawing::Point(110, 57);

this->input\_email->MaxLength = 254;

this->input\_email->Name = L"input\_email";

this->input\_email->Size = System::Drawing::Size(290, 32);

this->input\_email->TabIndex = 5;

this->input\_email->TabStop = false;

this->input\_email->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_email->TextChanged += gcnew System::EventHandler(this, &Registration::common\_contacts\_sufficient\_input\_check);

this->input\_email->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::keyboard\_focus\_switch\_check);

this->input\_email->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

//

// label\_email

//

this->label\_email->AutoSize = true;

this->label\_email->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_email->Location = System::Drawing::Point(14, 60);

this->label\_email->Name = L"label\_email";

this->label\_email->Size = System::Drawing::Size(105, 24);

this->label\_email->TabIndex = 1000;

this->label\_email->Text = L"Эл. почта:";

//

// registration\_company

//

this->registration\_company->Controls->Add(this->label\_year);

this->registration\_company->Controls->Add(this->label\_month);

this->registration\_company->Controls->Add(this->input\_website);

this->registration\_company->Controls->Add(this->label\_y);

this->registration\_company->Controls->Add(this->input\_com\_year);

this->registration\_company->Controls->Add(this->input\_com\_month);

this->registration\_company->Controls->Add(this->label\_website);

this->registration\_company->Controls->Add(this->label\_est);

this->registration\_company->Controls->Add(this->input\_legal\_form);

this->registration\_company->Controls->Add(this->label\_company\_name);

this->registration\_company->Controls->Add(this->input\_company\_name);

this->registration\_company->Font = (gcnew System::Drawing::Font(L"Roboto", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->registration\_company->Location = System::Drawing::Point(39, 141);

this->registration\_company->Name = L"registration\_company";

this->registration\_company->Size = System::Drawing::Size(410, 172);

this->registration\_company->TabIndex = 1001;

this->registration\_company->TabStop = false;

this->registration\_company->Text = L"Данные компании";

//

// label\_year

//

this->label\_year->AutoSize = true;

this->label\_year->Font = (gcnew System::Drawing::Font(L"Roboto", 7.8F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_year->Location = System::Drawing::Point(315, 90);

this->label\_year->Name = L"label\_year";

this->label\_year->Size = System::Drawing::Size(42, 15);

this->label\_year->TabIndex = 1009;

this->label\_year->Text = L"гойда";

//

// label\_month

//

this->label\_month->AutoSize = true;

this->label\_month->Font = (gcnew System::Drawing::Font(L"Roboto", 7.8F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_month->Location = System::Drawing::Point(206, 90);

this->label\_month->Name = L"label\_month";

this->label\_month->Size = System::Drawing::Size(53, 15);

this->label\_month->TabIndex = 1008;

this->label\_month->Text = L"месяца";

//

// input\_website

//

this->input\_website->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_website->Location = System::Drawing::Point(88, 109);

this->input\_website->Name = L"input\_website";

this->input\_website->Size = System::Drawing::Size(280, 32);

this->input\_website->TabIndex = 2;

this->input\_website->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_website->TextChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_company\_registration\_check);

this->input\_website->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::keyboard\_focus\_switch\_check);

this->input\_website->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

//

// label\_y

//

this->label\_y->AutoSize = true;

this->label\_y->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_y->Location = System::Drawing::Point(374, 62);

this->label\_y->Name = L"label\_y";

this->label\_y->Size = System::Drawing::Size(23, 24);

this->label\_y->TabIndex = 1006;

this->label\_y->Text = L"г.";

//

// input\_com\_year

//

this->input\_com\_year->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_com\_year->Location = System::Drawing::Point(294, 59);

this->input\_com\_year->MaxLength = 4;

this->input\_com\_year->Name = L"input\_com\_year";

this->input\_com\_year->Size = System::Drawing::Size(75, 32);

this->input\_com\_year->TabIndex = 1;

this->input\_com\_year->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_com\_year->TextChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_company\_registration\_check);

this->input\_com\_year->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::only\_digits);

this->input\_com\_year->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

//

// input\_com\_month

//

this->input\_com\_month->AutoCompleteMode = System::Windows::Forms::AutoCompleteMode::Append;

this->input\_com\_month->AutoCompleteSource = System::Windows::Forms::AutoCompleteSource::ListItems;

this->input\_com\_month->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->input\_com\_month->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_com\_month->FormattingEnabled = true;

this->input\_com\_month->Items->AddRange(gcnew cli::array< System::Object^ >(12) {

L"января", L"февраля", L"марта", L"апреля",

L"мая", L"июня", L"июля", L"августа", L"сентября", L"октября", L"ноября", L"декабря"

});

this->input\_com\_month->Location = System::Drawing::Point(160, 57);

this->input\_com\_month->MaxLength = 4;

this->input\_com\_month->Name = L"input\_com\_month";

this->input\_com\_month->Size = System::Drawing::Size(129, 32);

this->input\_com\_month->TabIndex = 1004;

this->input\_com\_month->SelectedValueChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_company\_registration\_check);

this->input\_com\_month->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

//

// label\_website

//

this->label\_website->AutoSize = true;

this->label\_website->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_website->Location = System::Drawing::Point(14, 112);

this->label\_website->Name = L"label\_website";

this->label\_website->Size = System::Drawing::Size(61, 24);

this->label\_website->TabIndex = 1003;

this->label\_website->Text = L"Сайт:";

//

// label\_est

//

this->label\_est->AutoSize = true;

this->label\_est->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_est->Location = System::Drawing::Point(14, 62);

this->label\_est->Name = L"label\_est";

this->label\_est->Size = System::Drawing::Size(121, 24);

this->label\_est->TabIndex = 1002;

this->label\_est->Text = L"На рынке с:";

//

// input\_legal\_form

//

this->input\_legal\_form->AutoCompleteMode = System::Windows::Forms::AutoCompleteMode::Append;

this->input\_legal\_form->AutoCompleteSource = System::Windows::Forms::AutoCompleteSource::ListItems;

this->input\_legal\_form->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->input\_legal\_form->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_legal\_form->FormattingEnabled = true;

this->input\_legal\_form->Items->AddRange(gcnew cli::array< System::Object^ >(8) {

L"ИП", L"ПК", L"ГУП", L"МУП", L"НКО", L"ООО",

L"ПАО", L"НПАО"

});

this->input\_legal\_form->Location = System::Drawing::Point(62, 22);

this->input\_legal\_form->MaxLength = 4;

this->input\_legal\_form->Name = L"input\_legal\_form";

this->input\_legal\_form->Size = System::Drawing::Size(84, 32);

this->input\_legal\_form->TabIndex = 1001;

this->input\_legal\_form->SelectedValueChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_company\_registration\_check);

this->input\_legal\_form->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

//

// label\_company\_name

//

this->label\_company\_name->AutoSize = true;

this->label\_company\_name->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label\_company\_name->Location = System::Drawing::Point(14, 28);

this->label\_company\_name->Name = L"label\_company\_name";

this->label\_company\_name->Size = System::Drawing::Size(55, 24);

this->label\_company\_name->TabIndex = 1000;

this->label\_company\_name->Text = L"Имя:";

//

// input\_company\_name

//

this->input\_company\_name->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->input\_company\_name->Location = System::Drawing::Point(148, 22);

this->input\_company\_name->MaxLength = 20;

this->input\_company\_name->Name = L"input\_company\_name";

this->input\_company\_name->Size = System::Drawing::Size(220, 32);

this->input\_company\_name->TabIndex = 0;

this->input\_company\_name->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

this->input\_company\_name->TextChanged += gcnew System::EventHandler(this, &Registration::sufficient\_input\_for\_company\_registration\_check);

this->input\_company\_name->KeyPress += gcnew System::Windows::Forms::KeyPressEventHandler(this, &Registration::only\_letters);

this->input\_company\_name->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &Registration::mouse\_focus\_switch\_check);

//

// button\_register

//

this->button\_register->Enabled = false;

this->button\_register->Font = (gcnew System::Drawing::Font(L"Roboto", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button\_register->Location = System::Drawing::Point(150, 467);

this->button\_register->Name = L"button\_register";

this->button\_register->Size = System::Drawing::Size(217, 52);

this->button\_register->TabIndex = 100;

this->button\_register->Text = L"Зарегистрироваться";

this->button\_register->UseVisualStyleBackColor = true;

this->button\_register->Click += gcnew System::EventHandler(this, &Registration::registration);

//

// Registration

//

this->AutoScaleDimensions = System::Drawing::SizeF(7, 15);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(492, 528);

this->Controls->Add(this->button\_register);

this->Controls->Add(this->button\_company);

this->Controls->Add(this->contacts);

this->Controls->Add(this->button\_individual);

this->Controls->Add(this->label\_no\_account);

this->Controls->Add(this->label\_kalimera);

this->Controls->Add(this->label\_registration);

this->Controls->Add(this->registration\_company);

this->Controls->Add(this->registration\_individual);

this->Font = (gcnew System::Drawing::Font(L"Roboto", 8, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->MaximumSize = System::Drawing::Size(510, 575);

this->MinimumSize = System::Drawing::Size(510, 575);

this->Name = L"Registration";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"Регистрация";

this->registration\_individual->ResumeLayout(false);

this->registration\_individual->PerformLayout();

this->contacts->ResumeLayout(false);

this->contacts->PerformLayout();

this->registration\_company->ResumeLayout(false);

this->registration\_company->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private:

void registration(System::Object^ sender, System::EventArgs^ e)

{

/// Common between 'Individual' and 'Company'.

std::wstring email = to\_std\_wstring(input\_email->Text);

std::wstring extra\_contacts = to\_std\_wstring(input\_extra->Text);

std::wstring carrier\_code = to\_std\_wstring(input\_carrier\_code->Text);

carrier\_code = erase\_at(carrier\_code, 0); // '('

carrier\_code = erase\_at(carrier\_code, 3); // ')', taking the previous removal and the resulting shift of positions into account

std::wstring body = to\_std\_wstring(input\_phone\_number\_body->Text);

body = erase\_at(body, 3); // '-'

body = erase\_at(body, 5); // '-'

Phone\_Number phone\_number(to\_std\_wstring(input\_country\_code->Text), carrier\_code, body);

// Saving an individual

if (button\_individual->Checked)

{

Individual\_Name name(to\_std\_wstring(input\_name->Text), to\_std\_wstring(input\_surname->Text), to\_std\_wstring(input\_patronym->Text));

Date birth\_date(dstoi(input\_ind\_day->Text), input\_ind\_month->SelectedIndex + 1, dstoi(input\_ind\_year->Text));

Individual\* new\_user = new Individual(name, birth\_date, phone\_number, email, extra\_contacts);

user = new\_user;

}

// Saving a company

else if (button\_company->Checked)

{

std::wstring website = to\_std\_wstring(input\_website->Text);

Company\_Name name(to\_std\_wstring(input\_legal\_form->Text), to\_std\_wstring(input\_company\_name->Text));

Date birth\_date(01, input\_com\_month->SelectedIndex + 1, dstoi(input\_com\_year->Text));

Company\* new\_user = new Company(name, birth\_date, phone\_number, email, website, extra\_contacts);

user = new\_user;

}

save(user, USER\_SAVEFILE\_NAME);

this->Close();

}

#pragma region ========== Individual ==========

void pick\_as\_individual(System::Object^ sender, System::EventArgs^ e)

{

set\_focus\_params\_individual();

node\_manipulations\_individual();

sufficient\_input\_for\_individual\_registration\_check(nullptr, nullptr);

}

void set\_focus\_params\_individual()

{

FOCUSABLE\_NODES->Resize(FOCUSABLE\_NODES, 9);

FOCUSABLE\_NODES[0] = input\_surname;

FOCUSABLE\_NODES[1] = input\_name;

FOCUSABLE\_NODES[2] = input\_patronym;

FOCUSABLE\_NODES[3] = input\_ind\_day;

FOCUSABLE\_NODES[4] = input\_ind\_year;

FOCUSABLE\_NODES[5] = input\_carrier\_code;

FOCUSABLE\_NODES[6] = input\_phone\_number\_body;

FOCUSABLE\_NODES[7] = input\_email;

FOCUSABLE\_NODES[8] = input\_extra;

MAX\_FOCUS = 8;

focus = 0;

FOCUSABLE\_NODES[0]->Select();

input\_carrier\_code->TabIndex = 5;

input\_phone\_number\_body->TabIndex = 6;

}

void node\_manipulations\_individual()

{

label\_if\_you\_have\_email->Show();

registration\_individual->BringToFront();

}

void sufficient\_input\_for\_individual\_registration\_check(System::Object^ sender, System::EventArgs^ e)

{

button\_register->Enabled = (

input\_surname->Text->Length >= 2 &&

input\_name->Text->Length >= 2 &&

input\_ind\_day->Text->Length > 0 &&

input\_ind\_month->SelectedIndex != -1 &&

input\_ind\_year->Text->Length == 4 &&

input\_carrier\_code->Text->Length == 5 &&

input\_phone\_number\_body->Text->Length == 9

);

}

#pragma endregion

#pragma region ========== Company ==========

void pick\_as\_company(System::Object^ sender, System::EventArgs^ e)

{

set\_focus\_params\_company();

node\_manipulations\_company();

sufficient\_input\_for\_company\_registration\_check(nullptr, nullptr);

}

void set\_focus\_params\_company()

{

FOCUSABLE\_NODES->Resize(FOCUSABLE\_NODES, 7);

FOCUSABLE\_NODES[0] = input\_company\_name;

FOCUSABLE\_NODES[1] = input\_com\_year;

FOCUSABLE\_NODES[2] = input\_website;

FOCUSABLE\_NODES[3] = input\_carrier\_code;

FOCUSABLE\_NODES[4] = input\_phone\_number\_body;

FOCUSABLE\_NODES[5] = input\_email;

FOCUSABLE\_NODES[6] = input\_extra;

MAX\_FOCUS = 6;

focus = 0;

FOCUSABLE\_NODES[0]->Select();

input\_carrier\_code->TabIndex = 3;

input\_phone\_number\_body->TabIndex = 4;

}

void node\_manipulations\_company()

{

label\_if\_you\_have\_email->Hide();

registration\_company->BringToFront();

}

void sufficient\_input\_for\_company\_registration\_check(System::Object^ sender, System::EventArgs^ e)

{

button\_register->Enabled = (

input\_legal\_form->SelectedIndex > -1 &&

input\_company\_name->Text->Length > 0 &&

input\_com\_month->SelectedIndex > -1 &&

input\_com\_year->Text->Length == 4 &&

input\_email->Text->Length > 0 &&

input\_website->Text->Length > 0 &&

input\_carrier\_code->Text->Length == 5 &&

input\_phone\_number\_body->Text->Length == 9

);

}

#pragma endregion

#pragma region ========== Focus (Common) ==========

int focus = 0;

int MAX\_FOCUS;

array<Control^>^ FOCUSABLE\_NODES;

void next\_focus()

{

// Focus wrapping had been decided to be disabled.

if (++focus > MAX\_FOCUS) focus = MAX\_FOCUS;

FOCUSABLE\_NODES[focus]->Focus();

}

void prev\_focus()

{

// Focus wrapping had been decided to be disabled.

if (--focus < 0) focus = 0;

FOCUSABLE\_NODES[focus]->Focus();

}

/// Assumes every 'sender' is a 'TextBox^'. Watch out!

System::Void keyboard\_focus\_switch\_check(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e)

{

System::Char key = e->KeyChar;

if (key == ENTER)

next\_focus();

else

{

TextBox^ source = safe\_cast<TextBox^>(sender);

if (key == BACKSPACE && source->Text == "")

prev\_focus();

}

}

/// Assumes every 'sender' is a 'Control^' or is derived from it. Watch out!

void mouse\_focus\_switch\_check(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e)

{

Control^ source = safe\_cast<Control^>(sender);

focus = source->TabIndex;

}

#pragma endregion

#pragma region ========== Input limiting (Common) ==========

/// Assumes every 'sender' is a 'TextBox^'. Watch out!

void only\_digits(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e)

{

System::Char key = e->KeyChar;

TextBox^ source = safe\_cast<TextBox^>(sender);

if (! is\_digit(key))

e->Handled = true;

else if (source->Text->Length == source->MaxLength - 1) /// - 1 because the event is yet unhandled,

next\_focus(); /// so the actual size we care about hasn't been achieved

keyboard\_focus\_switch\_check(sender, e);

}

/// Assumes every 'sender' is a 'TextBox^'. Watch out!

void only\_letters(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e)

{

System::Char key = e->KeyChar;

TextBox^ source = safe\_cast<TextBox^>(sender);

if (! is\_letter(key))

e->Handled = true;

else if (source->Text->Length == source->MaxLength - 1) /// - 1 because the event is yet unhandled,

next\_focus(); /// so the actual size we care about hasn't been achieved

keyboard\_focus\_switch\_check(sender, e);

}

#pragma endregion

#pragma region ========== Input (Contacts) ==========

void common\_contacts\_sufficient\_input\_check(System::Object^ sender, System::EventArgs^ e)

{

if (button\_individual->Checked)

sufficient\_input\_for\_individual\_registration\_check(nullptr, nullptr);

else if (button\_company->Checked)

sufficient\_input\_for\_company\_registration\_check(nullptr, nullptr);

}

int \_carrier\_code\_previous\_length = 0;

System::Void on\_carrier\_code\_input(System::Object^ sender, System::EventArgs^ e)

{

// inserting '('

if (input\_carrier\_code->Text->Length == 1 && \_carrier\_code\_previous\_length == 0)

{

input\_carrier\_code->Text = "(" + input\_carrier\_code->Text;

input\_carrier\_code->SelectionStart = 2;

}

// inserting ')'

else if (input\_carrier\_code->Text->Length == 4 && \_carrier\_code\_previous\_length == 3)

{

input\_carrier\_code->Text += L')';

input\_carrier\_code->SelectionStart = 5;

next\_focus();

}

\_carrier\_code\_previous\_length = input\_carrier\_code->Text->Length;

common\_contacts\_sufficient\_input\_check(nullptr, nullptr);

}

unsigned int \_phone\_number\_body\_previous\_length = 0;

void on\_phone\_number\_body\_input(System::Object^ sender, System::EventArgs^ e)

{

// Inserting first '-'

if (input\_phone\_number\_body->Text->Length == 3 && \_phone\_number\_body\_previous\_length == 2)

{

input\_phone\_number\_body->Text += L'-';

input\_phone\_number\_body->SelectionStart = 4;

}

// Inserting second '-'

else if (input\_phone\_number\_body->Text->Length == 6 && \_phone\_number\_body\_previous\_length == 5)

{

input\_phone\_number\_body->Text += L'-';

input\_phone\_number\_body->SelectionStart = 7;

}

\_phone\_number\_body\_previous\_length = input\_phone\_number\_body->Text->Length;

common\_contacts\_sufficient\_input\_check(nullptr, nullptr);

}

#pragma endregion

};

}

#endif

#### Листинг кода datetime.hpp

#pragma once

#ifndef DATE\_TIME\_HPP

#define DATE\_TIME\_HPP

#include <time.h> /// time

#include <string> /// 'std::wstring'

#include <vector>

/// This file is inside of "Utils",

/// => we want to include any parts, we must call to them directly.

#include "strings.hpp"

#include "random.hpp"

/// Here be:

class Date;

Date \_get\_random\_date();

std::wstring get\_date();

Date get\_date\_as\_obj();

int get\_time();

const std::wstring MONTHS\_GENITIVE[12] = {

L"января", L"февраля",

L"марта", L"апреля", L"мая",

L"июня", L"июля", L"августа",

L"сентября", L"октября", L"ноября",

L"декабря"};

class Date

{

public:

Date()

{/\* default constructor for errors to shut up \*/}

/// Out-of-file initialization

Date(const std::wstring& \_raw\_date)

{

const std::vector<std::wstring> separated = separate(\_raw\_date);

day = \_wtoi(separated[0].c\_str());

month = \_wtoi(separated[1].c\_str());

year = \_wtoi(separated[2].c\_str());

}

/// Registration

Date(const unsigned int \_day, const unsigned int \_month, const unsigned int \_year)

: day(\_day), month(\_month), year(\_year)

{/\* could run some checks on input values \*/}

std::vector<std::wstring> as\_vector()

{

const std::vector<std::wstring> result = {std::to\_wstring(day), std::to\_wstring(month), std::to\_wstring(year)};

return result;

}

std::wstring dd\_mm\_yyyy(const std::wstring separator = L".")

{

std::wstring dd = std::to\_wstring(day);

if (dd.size() < 2)

dd = L"0" + dd;

std::wstring mm = std::to\_wstring(month);

if (mm.size() < 2)

mm = L"0" + mm;

std::wstring yyyy = std::to\_wstring(year);

return dd + separator + mm + separator + yyyy;

}

std::wstring dd\_month\_yyyy\_genitive()

{

std::wstring dd = std::to\_wstring(day);

if (dd.size() < 2)

dd = L"0" + dd;

return dd + L" " + MONTHS\_GENITIVE[month - 1] + L" " + std::to\_wstring(year) + L"г.";

}

std::wstring month\_yyyy\_genitive()

{

return MONTHS\_GENITIVE[month - 1] + L" " + std::to\_wstring(year) + L"г.";

}

int years\_since(Date since = get\_date\_as\_obj())

{

bool day\_month\_passed;

if (since.month > this->month)

day\_month\_passed = true;

else if (since.month == this->month)

{

if (since.day >= this->day)

day\_month\_passed = true;

else

day\_month\_passed = false;

}

else if (since.month < this->month)

return day\_month\_passed = false;

return since.year - this->year - ! day\_month\_passed;

}

private:

unsigned int day, month, year;

};

/// TODO: lower and upper bound; random date in x months, etc

Date \_get\_random\_date()

{

int day = randint(1, 28);

int month = randint(1, 12);

int year = randint(1970, 2007);

return Date(day, month, year);

}

std::wstring get\_date()

{

/// Copied from https://stackoverflow.com/a/68124506/15540979

time\_t rawtime;

struct tm timeinfo;

wchar\_t buffer[20];

time(&rawtime);

localtime\_s(&timeinfo, &rawtime);

wcsftime(buffer, 20, L"%d.%m.%Y", &timeinfo);

return buffer;

}

Date get\_date\_as\_obj()

{

/// Copied from https://stackoverflow.com/a/68124506/15540979

time\_t rawtime;

struct tm timeinfo;

time(&rawtime);

localtime\_s(&timeinfo, &rawtime);

wchar\_t day[4];

wcsftime(day, 4, L"%d", &timeinfo);

wchar\_t month[4];

wcsftime(month, 4, L"%m", &timeinfo);

wchar\_t year[8];

wcsftime(year, 8, L"%Y", &timeinfo);

Date date(\_wtoi(day), \_wtoi(month), \_wtoi(year));

return date;

}

int get\_time()

{

/// Copied from https://stackoverflow.com/a/68124506/15540979

time\_t rawtime;

struct tm timeinfo;

wchar\_t buffer[20];

time(&rawtime);

localtime\_s(&timeinfo, &rawtime);

wcsftime(buffer, 20, L"%H%M%S", &timeinfo);

return std::stoi(buffer);

}

#endif

#### Листинг кода files.hpp

#pragma once

#ifndef FILES\_HPP

#define FILES\_HPP

#include <fstream>

#include <string>

#include <vector>

/// Here be:

bool is\_empty(const std::wstring& address);

std::vector<std::wstring> to\_string\_bundle(std::wifstream& file, const unsigned int expected\_size = 0);

// Copied from https://stackoverflow.com/a/2390938/15540979

bool is\_empty(const std::wstring& address)

{

std::wifstream file(address); /// Loading file like that to not cut off character 1

return file.peek() == std::wifstream::traits\_type::eof();

}

/// NOTE: function assumes the file has been checked

std::vector<std::wstring> to\_string\_bundle(std::wifstream& file, const unsigned int expected\_size/\* = 0\*/)

{

std::wstring file\_line;

std::vector<std::wstring> raw\_data;

raw\_data.reserve(expected\_size);

while(getline(file, file\_line))

raw\_data.push\_back(file\_line);

return raw\_data;

}

#endif

#### Листинг кода messagebox\_wrapper.hpp

#pragma once

#ifndef MESSAGEBOX\_WRAPPER\_HPP

#define MESSAGEBOX\_WRAPPER\_HPP

#include <string>

#include "strings.hpp"

using namespace System::Windows::Forms;

/// Here be:

void show\_info (System::String^ info, const std::wstring& heading = L"Информация");

void show\_info (const std::wstring& info, const std::wstring& heading = L"Информация");

void show\_warning(const std::wstring& warning, const std::wstring& heading = L"Внимание!");

void show\_error (const std::wstring& error, const std::wstring& heading = L"Ошибка");

void show\_info(System::String^ info, const std::wstring& heading/\* = L"Информация"\*/)

{

MessageBox::Show(info, to\_dotnet\_string(heading), MessageBoxButtons::OK, MessageBoxIcon::Information);

}

void show\_info(const std::wstring& info, const std::wstring& heading/\* = L"Информация"\*/)

{

MessageBox::Show(to\_dotnet\_string(info), to\_dotnet\_string(heading), MessageBoxButtons::OK, MessageBoxIcon::Information);

}

void show\_warning(const std::wstring& warning, const std::wstring& heading/\* = L"Внимание!"\*/)

{

MessageBox::Show(to\_dotnet\_string(warning), to\_dotnet\_string(heading), MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

void show\_error(const std::wstring& error, const std::wstring& heading/\* = L"Ошибка"\*/)

{

MessageBox::Show(to\_dotnet\_string(error), to\_dotnet\_string(heading), MessageBoxButtons::OK, MessageBoxIcon::Error);

}

#endif

#### Листинг кода random.hpp

#pragma once

#ifndef RANDOM\_HPP

#define RANDOM\_HPP

#include <time.h> // time-based seed generation

/// Here be:

int randint(const unsigned int min, const unsigned int max);

int randint(const unsigned int max);

bool randint\_percent(const unsigned int percent);

unsigned int \_seed = time(NULL);

/// [min, max]. I DON'T THINK it supports negatives

int randint(const unsigned int min, const unsigned int max)

{

if (max < min) return -1;

srand(++\_seed);

return rand() % (max+1 - min) + min;

}

/// [0, max]. Does NOT support negatives!

int randint(const unsigned int max)

{

if (max < 0) return -1;

srand(++\_seed);

return rand() % (max+1);

}

/// Returns true if a percent% chance is hit

bool randint\_percent(const unsigned int percent)

{

if (percent > 100) return true;

return randint(1, 100) <= percent;

}

#endif

#### Листинг кода strings.hpp

#pragma once

#ifndef STRINGS\_HPP

#define STRINGS\_HPP

#include <msclr/marshal\_cppstd.h> // string type magic

#include <string> // 'std::wstring'

#include <sstream>

#include <unordered\_map> // mapping chars for transliteration

#include <vector>

#include <Windows.h>

using namespace System; // all things CLR

#define BACKSPACE (char)8

#define ENTER '\r'

std::wstring erase\_at(const std::wstring& target, const unsigned int index);

bool is\_digit (const System::Char target);

bool is\_letter(System::Char target, const std::wstring additional\_symbols = L"-' ё");

wchar\_t lower(wchar\_t target);

std::wstring lower(std::wstring target);

std::vector<std::wstring> separate(const std::wstring& raw\_data);

int dstoi(System::String^ target);

System::String^ to\_dotnet\_string(const std::wstring& target);

System::String^ to\_dotnet\_string(const int target);

std::wstring to\_std\_wstring(System::String^ target);

std::wstring translit\_CtoL(const std::wstring& target);

/// Because 'wstring::erase' simply refused to work.

std::wstring erase\_at(const std::wstring& target, const unsigned int index)

{

std::wstring result = L"";

for (int i = 0; i < target.size(); i++)

{

if (i == index)

continue;

result += target[i];

}

return result;

}

bool is\_digit(const System::Char target)

{

return (target >= L'0' && target <= L'9')

|| (target == BACKSPACE);// || target == ENTER); // ENTER is used to switch input focus,

// thus \*evilly\* MUST NOT BE TOLERATED

}

// I suppose 'System::Char' is just 'wchar' without directly stating it. Gods, I hate .NET

bool is\_letter(System::Char target, const std::wstring additional\_symbols/\* = L"-' ё"\*/)

{

target = lower(target);

return (target >= L'а' && target <= L'я')

|| (target >= L'a' && target <= L'z')

|| (additional\_symbols.find(target) != std::wstring::npos)

|| (target == BACKSPACE);// || target == ENTER);

}

wchar\_t lower(wchar\_t target)

{

if (target < L'а' && target >= L'А') // ru-RU

return target - (L'А' - L'а');

else if (target < L'a' && target >= L'A') // en-EN

return target - (L'A' - L'a');

else

return target;

}

std::wstring lower(std::wstring target)

{

for (int i = 0; i < target.size(); i++)

target[i] = lower(target[i]);

return target;

}

std::vector<std::wstring> separate(const std::wstring& raw\_data)

{

std::vector<std::wstring> result;

std::wstringstream raw\_data\_sstr;

raw\_data\_sstr.str(raw\_data);

std::wstring word;

while(raw\_data\_sstr >> word)

result.push\_back(word);

return result;

}

int dstoi(System::String^ target)

{

std::wstring middleman = to\_std\_wstring(target);

return \_wtoi(middleman.c\_str());

}

// Copied from: https://stackoverflow.com/a/1405251/15540979

System::String^ to\_dotnet\_string(const std::wstring& target)

{

return msclr::interop::marshal\_as<System::String^>(target);

}

/// 'int' to .NET string

System::String^ to\_dotnet\_string(const int target)

{

std::wstring middleman = std::to\_wstring(target);

return to\_dotnet\_string(middleman);

}

// Copied from: https://stackoverflow.com/a/1405251/15540979

std::wstring to\_std\_wstring(System::String^ target) // System::String^ doesn't like to be const

{

return msclr::interop::marshal\_as<std::wstring>(target);

}

const std::unordered\_map<wchar\_t, std::wstring> \_char\_transliterations {

{L'а', L"a"}, {L'б', L"b"}, {L'в', L"v"}, {L'г', L"g"}, {L'д', L"d"}, {L'е', L"e"}, {L'ё', L"yo"}, {L'ж', L"zh"},

{L'з', L"z"}, {L'и', L"i"}, {L'й', L"y"}, {L'к', L"k"}, {L'л', L"l"}, {L'м', L"m"}, {L'н', L"n"}, {L'о', L"o"},

{L'п', L"p"}, {L'р', L"r"}, {L'с', L"s"}, {L'т', L"t"}, {L'у', L"u"}, {L'ф', L"f"}, {L'х', L"kh"}, {L'ц', L"ts"},

{L'ч', L"ch"}, {L'ш', L"sh"}, {L'щ', L"sch"}, {L'ъ', L"y"}, {L'ы', L"y"}, {L'ь', L""}, {L'э', L"e"}, {L'ю', L"yu"},

{L'я', L"ya"}, {L' ', L"\_"}

};

std::wstring translit\_CtoL(const std::wstring& target)

{

/// We could make edits directly to 'target',

/// but it was causing very hard-to-catch errors.

std::wstring result = L"";

for (int i = 0; i < target.size(); i++)

{

/// Lowering happens because

/// 1) translit is only ever needed while lowering,

/// 2) I'm lazy

wchar\_t c = lower(target[i]);

if (\_char\_transliterations.find(c) == \_char\_transliterations.end())

result += c;

else

result += \_char\_transliterations.at(c);

}

return result;

}

#endif

#### Листинг кода main.cpp

#include <string>

#include <Windows.h>

#include "Menus/Main\_Menu/main\_menu.hpp"

#include "Menus/Registration/registration.hpp"

#include "utils.hpp"

using namespace Agora;

using namespace System::Windows::Forms;

/// Note: we are ABSOLUTELY not ready for user-made listings. There are SO many checks in place preventing that.

[System::STAThread]

int WINAPI wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nShowCmd)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

User\* user = load(USER\_SAVEFILE\_NAME);

if (user == nullptr) /// Not registered.

{

Application::Run(gcnew Registration(user));

user = load(USER\_SAVEFILE\_NAME); /// loading saved data

}

if (user == nullptr) /// Registration didn't work, or user closed the window.

return 0;

load\_predefined\_companies();

load\_saved\_individuals();

Application::Run(gcnew Main\_Menu(user));

return 0;

}

#### Листинг кода save\_load.hpp

#pragma once

#ifndef SAVE\_LOAD\_HPP

#define SAVE\_LOAD\_HPP

#include <codecvt> /// Locales

#include <fstream> /// File reading and writing

#include <string> /// 'std::wstring>

#include <vector> /// vectors of 'std::wstring'

#include "Classes/listing.hpp"

#include "Classes/user.hpp"

#include "utils.hpp"

/// This file is for saving and loading 'Individuals's.

/// Companies are predefined instead of randomly generated.

/// [!!!] FILE MANIPULATION is in 'Utils/files.hpp'.

/// Here be:

const std::wstring SAVEFILES\_DIRECTORY = L"./SaveData/";

const std::string USER\_SAVEFILE\_NAME\_S = "user.txt";

const std::wstring USER\_SAVEFILE\_NAME = L"user.txt";

const std::wstring USER\_LISTINGS\_FILE\_NAME = L"user\_listings.txt";

const std::wstring SAVED\_INDIVIDUALS\_FILE\_NAME = L"listing\_individuals.txt";

User\* load(const std::wstring address);

void save(User\* user, std::wstring address = L"", std::vector<Listing> active\_listings = {});

std::vector<Listing> load\_listings(User\* user);

void load\_predefined\_companies();

void load\_saved\_individuals();

std::vector<User\*> PREDEFINED\_COMPANIES;

std::vector<User\*> SAVED\_INDIVIDUALS;

/// POSSIBLE MEMORY LEAK: No time to deal with this pointer magic.

User\* load(const std::wstring filename)

{

std::wifstream savefile(SAVEFILES\_DIRECTORY + filename);

/// Empty check

if (is\_empty(SAVEFILES\_DIRECTORY + filename))

{

savefile.close();

return nullptr;

}

// Copied from: https://stackoverflow.com/a/3950840/15540979

std::locale loc(std::locale::classic(), new std::codecvt\_utf8<wchar\_t>);

savefile.imbue(loc);

// The copied code deals with locales, enabling a full save into file without errors. I spent so much time!

/// Account type check

std::wstring account\_type;

getline(savefile, account\_type);

if (account\_type != L"individual" && account\_type != L"company")

{

show\_error((std::wstring)L"'" + filename + L"' - неправильный тип аккаунта '" + account\_type + L"'");

savefile.close();

return nullptr;

}

/// Reading the rest of the data

std::vector<std::wstring> raw\_data = to\_string\_bundle(savefile, 7);

savefile.close();

/// Creating a new 'User\*'

/// POSSIBLE MEMORY LEAK: No time to deal with this pointer magic.

/// WARNING: change of order => this breaks. Wish I had time to rewrite this to JSON.

/// Common variables

const std::wstring \_birth\_date = raw\_data[0], /\* Note how we read from 0, not 1: \*/

\_name = raw\_data[1], /\* account type didn't get into raw data, \*/

\_phone\_number = raw\_data[2], /\* because it's already been read. \*/

email = raw\_data[3],

extra\_contacts = raw\_data[4];

/// Absolutely no checks in place. This shit can break in so many ways

Date birth\_date(\_birth\_date);

Phone\_Number phone\_number(separate(\_phone\_number));

if (account\_type == L"individual")

{

Individual\_Name name(separate(\_name));

return new Individual(name, birth\_date, phone\_number, email, extra\_contacts);

}

else if (account\_type == L"company")

{

std::wstring website = raw\_data[5];

Company\_Name name(separate(\_name));

return new Company(name, birth\_date, phone\_number, email, website, extra\_contacts);

}

show\_error(L"Аккаунт не создан по неизвестной причине.");

return nullptr; /// Something that under NO circumstance should happen

}

/// 'User\*' instead of raw text as a protection from saving jackshit

void save(User\* user, std::wstring filename/\* = L""\*/, std::vector<Listing> active\_listings/\* = {}\*/)

{

if (filename == L"")

filename = user->name->as\_filename();

std::wofstream savefile(SAVEFILES\_DIRECTORY + filename);

// Copied from: https://stackoverflow.com/a/3950840/15540979

std::locale loc(std::locale::classic(), new std::codecvt\_utf8<wchar\_t>);

savefile.imbue(loc);

// The copied code deals with locales, enabling a full save into file without errors. I spent so much time!

savefile << user->serialize();

savefile.close();

if (active\_listings.empty())

return;

std::wofstream savefile\_listings(SAVEFILES\_DIRECTORY + USER\_LISTINGS\_FILE\_NAME);

savefile\_listings.imbue(loc);

std::wofstream savefile\_individuals(SAVEFILES\_DIRECTORY + SAVED\_INDIVIDUALS\_FILE\_NAME);

savefile\_individuals.imbue(loc);

for (Listing& listing : active\_listings)

{

savefile\_listings << listing.serialize() << L'\n';

if ((listing.contractor != user) && (typeid(\*listing.contractor) == typeid(Individual)))

savefile\_individuals << listing.contractor->serialize() << L'\n';

else if ((listing.customer != user) && (typeid(\*listing.customer) == typeid(Individual)))

savefile\_individuals << listing.customer->serialize() << L'\n';

}

savefile\_listings.close();

savefile\_individuals.close();

}

std::vector<Listing> load\_listings(User\* user)

{

if (PREDEFINED\_COMPANIES.empty())

{

show\_error(L"Predefined companies need to be loaded before listings.");

return {};

}

std::wifstream savefile(SAVEFILES\_DIRECTORY + USER\_LISTINGS\_FILE\_NAME);

/// Empty check

if (is\_empty(SAVEFILES\_DIRECTORY + USER\_LISTINGS\_FILE\_NAME))

{

savefile.close();

return {};

}

// Copied from: https://stackoverflow.com/a/3950840/15540979

std::locale loc(std::locale::classic(), new std::codecvt\_utf8<wchar\_t>);

savefile.imbue(loc);

// The copied code deals with locales, enabling a full save into file without errors. I spent so much time!

std::vector<std::wstring> raw\_data = to\_string\_bundle(savefile);

savefile.close();

std::vector<Listing> result;

for (int i = 0; i < raw\_data.size(); i++)

{

if (raw\_data[i] == L"")

continue;

/// Reading raw data in segments of 5

std::wstring name = raw\_data[i++];

unsigned int payment\_hr = stoi(raw\_data[i++]);

unsigned int hrs = stoi(raw\_data[i++]);

User \*contractor = nullptr, \*customer = nullptr, \*author = nullptr;

std::wstring contractor\_path = raw\_data[i++];

if (contractor\_path == L"user.txt")

contractor = user;

// I'm sure I can refactor this.

else

{

for (User\* company : PREDEFINED\_COMPANIES)

{

if (company->name->as\_filename() == contractor\_path)

contractor = company;

}

if (contractor == nullptr) /// contractor NOT found in companies

{

for (User\* individual : SAVED\_INDIVIDUALS)

{

if (individual->name->as\_filename() == contractor\_path)

contractor = individual;

}

}

author = contractor;

}

std::wstring customer\_path = raw\_data[i];

if (customer\_path == L"user.txt")

customer = user;

else

{

for (User\* company : PREDEFINED\_COMPANIES)

{

if (company->name->as\_filename() == customer\_path)

customer = company;

}

if (contractor == nullptr) /// customer NOT found in companies

{

for (User\* individual : SAVED\_INDIVIDUALS)

{

if (individual->name->as\_filename() == customer\_path)

customer = individual;

}

}

author = customer;

}

Listing new\_listing(name, hrs, payment\_hr, contractor, customer, author);

result.push\_back(new\_listing);

}

return result;

}

/// Wish I could do that via scanning all the files in a folder.

/// Sadly, 'filesystem' gave me some hellish errors...

const std::wstring \_PREDEFINED\_COMPANY\_SAVEPATHS[10] = {

L"agroprom.txt", L"ekoresurs.txt", L"energoplyus.txt", L"finansovyy\_alyans.txt", L"innovatsii\_buduschego.txt",

L"medtekh.txt", L"softrazrabotka.txt", L"stroymaster.txt", L"tekhnoprogress.txt", L"tsifrovye\_resheniya.txt"

};

/// POSSIBLE MEMORY LEAK

void load\_predefined\_companies()

{

PREDEFINED\_COMPANIES.reserve(10);

for (std::wstring filename : \_PREDEFINED\_COMPANY\_SAVEPATHS)

{

filename = L"/Predefined\_Companies/" + filename;

User\* new\_company = load(filename);

PREDEFINED\_COMPANIES.push\_back(new\_company);

}

}

void load\_saved\_individuals()

{

std::wifstream savefile(SAVEFILES\_DIRECTORY + SAVED\_INDIVIDUALS\_FILE\_NAME);

/// Empty check

if (is\_empty(SAVEFILES\_DIRECTORY + SAVED\_INDIVIDUALS\_FILE\_NAME))

{

savefile.close();

return;

}

// Copied from: https://stackoverflow.com/a/3950840/15540979

std::locale loc(std::locale::classic(), new std::codecvt\_utf8<wchar\_t>);

savefile.imbue(loc);

// The copied code deals with locales, enabling a full save into file without errors. I spent so much time!

std::vector<std::wstring> raw\_data = to\_string\_bundle(savefile);

savefile.close();

std::wstring birth\_date, name, phone\_number, email, extra\_contacts;

for (int i = 0; i < raw\_data.size(); i++)

{

if (raw\_data[i] == L"")

continue;

++i; /// accounting for 'individual' line in every serialization

/// Reading raw data in segments of 5

birth\_date = raw\_data[i++];

name = raw\_data[i++];

phone\_number = raw\_data[i++],

email = raw\_data[i++],

extra\_contacts = raw\_data[i++];

Date birth\_date(birth\_date);

Phone\_Number phone\_number(separate(phone\_number));

Individual\_Name name(separate(name));

SAVED\_INDIVIDUALS.push\_back(new Individual(name, birth\_date, phone\_number, email, extra\_contacts));

}

}

#endif

#### Листинг кода utils.hpp

#pragma once

#ifndef UTILS\_HPP

#define UTILS\_HPP

/// Bundle include of:

#include "Utils/files.hpp"

#include "Utils/datetime.hpp"

#include "Utils/messagebox\_wrapper.hpp"

#include "Utils/random.hpp"

#include "Utils/strings.hpp"

/// Do NOT write any functionality for this file, because

/// 'Utils' headers sometimes need to intertwine.

#endif