ДЕПАРТАМЕНТ ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ ТОМСКОЙ ОБЛАСТИ ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ «ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Специальность 09.02.07 «Информационные системы и программирование»

КУРСОВОЙ ПРОЕКТ  
 РАЗРАБОТКА И ТЕСТИРОВАНИЕ  
 ИНФОРМАЦИОННОЙ СИСТЕМЫ «СТРЕЛОК»

Пояснительная записка  
к курсовому проекту  
КП.09.02.07.602.11.2022.ПЗ

Студент  
«\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В.Коровин  
Руководитель  
«\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А.Антипов

Томск 2022

СОДЕРЖАНИЕ

[Введение 3](#_Toc122583436)

[1 ОБЩАЯ ЧАСТЬ 4](#_Toc122583437)

[1.1 Анализ предметной области 4](#_Toc122583438)

[1.2 Выбор средств и сред разработок 6](#_Toc122583439)

[2 СПЕЦИАЛЬНАЯ ЧАСТЬ 7](#_Toc122583440)

[2.1 Описание требований к информационной системе 7](#_Toc122583441)

[2.2 Диаграмма вариантов использования 8](#_Toc122583442)

[2.3 Диаграмма состояний 9](#_Toc122583443)

[2.4 Схема данных 10](#_Toc122583444)

[2.5 Пользовательские сценарии 11](#_Toc122583445)

[2.6 Прототипы основных интерфейсов 12](#_Toc122583446)

[3 ЗАКЛЮЧЕНИЕ 15](#_Toc122583447)

[3.1 Приложение A. Листинг окон программы 15](#_Toc122583448)

[3.2 Приложение В. Листинг кода программы 27](#_Toc122583449)

[3.2.1 Модуль общей формы МенюПреподавателя: 27](#_Toc122583450)

[3.2.2 Модуль общей формы МенюУченика: 29](#_Toc122583451)

[3.2.3 Модуль общей формы МенюАдмина 30](#_Toc122583452)

[3.2.4 Модуль формы элемента справочника Теория 32](#_Toc122583453)

[3.2.5 Запрос схемы компоновки данных отчета Расписание 34](#_Toc122583454)

[3.2.6 Запрос схемы компоновки данных отчета Успеваемость 35](#_Toc122583455)

[3.2.7 Модуль формы обработки Тест 35](#_Toc122583456)

[3.2.8 Модуль формы MainFrom обработки СменаПользователя: 42](#_Toc122583457)

Введение

Все люди могут охотиться в стране на определенных животных в периоды, когда можно и не повредит популяции животных. Но из-за законодательства страны официальный доступ к стрелковому оружию запрещен, но его можно «разблокировать», пройдя обучение на соответствующий тип оружия (травматическое, охотничье). Сейчас можно пройти обучение в любой стрелковой школе, но для этого нужно иметь 18+ возраст и пару медицинских документов, подтверждающие отсутствия зависимости от наркотиков и психического здоровья. Но все могут пройти обучение только на травматическое и охотничье, а на доступ к нарезному будет доступ после достижения стажа владением охотничьем оружием и прохождении обучении на этот тип оружия. Обучение представляет собой теорию и практику, которую проходят в тире или на полигонах. Обучение в таких школах доступно для мирного населения, но обучение можно пройти в армии.

ОБЩАЯ ЧАСТЬ

* 1. Анализ предметной области

Приложение будет автоматизировать:

* Регистрацию и авторизацию
* Прикрепление документов о мед осмотре и наркотическом осмотре
* Просмотр и изучение теоритического материала на гладкоствольное оружие или травматическое (видео или статьи)
* Прохождение теста после изучения всего материала
* Отправка уведомления о прохождении теста человеком
* Отправка электронного документа руководителю для дальнейшего подписания  
  (вся информация, которую человек ввел при регистрации и то, что он прошел теорию)
* Выбор дня, в который человек готов пройти практическую часть обучения
* Сохранения документов и сведений о человеке прошедшем обучении
* Просмотр, кто прошел обучение
* Возможность связаться с человеком через приложение для уточнении продолжения

Пользователь после регистрации и авторизации может ознакомиться с теорией, после просмотра ролика или прочтения статьи пользователь может на галочку, что пользователь прошел, изучил этот этап, и не мог подглянуть во время теоритического экзамена, после прохождении всех этапов теории, человек может пройти тест, который отсылает документ о успешном прохождении теста с указанием данных человека руководителю. После прохождения теста, если неудачно, то человеку открывается теория снова, но через некоторое время и возможность пройти тест снова после прохождения теории снова, если удачно то человеку открывается окно для переписки с руководителей

Анализ целевой аудитории:

Люди возрастом 18+, которые хотят обучение для получения официальной возможности получить после 21 года лицензию на оружие и права владеть оружие.

Анализ конкурентов:

1. <https://akademinex.ru/go>

+:

Можно записаться на обучение

Пройти экзамен или изучить вопросы экзамена

Номер телефона,адрес школы, адрес почты для связи

-

Отсутствие доступной теории для подготовки к прохождению экзамена

Отсутствие окна для онлайн-связи с руководителем и менеджером

Отсутствие обучающих видео для обучения

1. <https://tomtit.tomsk.ru/page/39/shkola-ohrannikov>

+

Ссылка на вопросы для подготовки к тесту

Контакты для связи

Список требуемых документов

-

Отсутствие окна для онлайн связи

Окна для изучении теории

1. <http://www.bert-tir.ru/market/napolnenie_kontentom/>

+

Документ для обучения на охотничье оружие

Вопросы для проверки знаний об обучении

Контакты для связи

-

Отсутствие видео для прохождения обучения

Отсутствие окна для онлайн связи

Отсутствие онлайн подачи заявки

* 1. Выбор средств и сред разработок

Платформа «1С:Предприятие 8.3» — набор инструментов для создания бизнес-приложений и среда их выполнения. Это большой (более десятка миллионов строк кода) проект на С++, Java и JavaScript. Над ним трудятся десятки программистов, одновременно разрабатывающие и поддерживающие до 10 различных версий продукта.

Платформа работает на различных версиях ОС и БД:

* ОС: Windows, Linux, macOS
* СУБД: MS SQL, PostgreSQL, IBM DB2, Oracle, файловая СУБД собственной разработки
* Мобильные ОС: Android, iOS, Windows

Было выбрано 1С:Предприятие 8.3 «Учебная версия» для разработки системы для курсовой работы, потому что платформа уже имеет различные объекты и позволяет не делать новые объекты, а работать уже с доступными. Еще одна причина выбора этой среды разработки – это достаточное простое изучение этого языка и платформы, но у этой среды есть свои минусы. Проверять систему буду проверять ручным тестированием, так как мало средств тестирования для проверки систем, сделанные на базе 1С. Большую часть из этих средств либо платные, либо устаревшие. Поэтому выбор был сделан в пользу ручного тестирования. Методологией для тестирования системы, которую я разрабатываю для курсовой работы, будет методология “Белый ящик”. Так как это методология подразумевает открытый код для разработчика.

СПЕЦИАЛЬНАЯ ЧАСТЬ

* 1. Описание требований к информационной системе

**Тест** –

1. Может пройти каждый пользователь, который зарегистрирован в системе.
2. После прохождения теста формируется документ о прохождении теста.

**Расписание** –

1. Отчет должен отображать расписание уроков, предметов и кабинетов, в которых проходят предметы
2. Редактировать и добавлять запись в расписание

**Успеваемость** –

1. Должна отображать оценки учеников
2. Возможность добавления записей оценок учеников

**Теория** –

1. Должна отображать текст и изображение для дополнения к тексту
2. Возможность добавления тем теории, описания и изображения

**СменаПользователя** –

1. Позволяет сменить сессию пользователя

**ПрошедшиеТесты** –

1. Формируется после прохождения теста
2. Отображение всех записей о прохождении теста

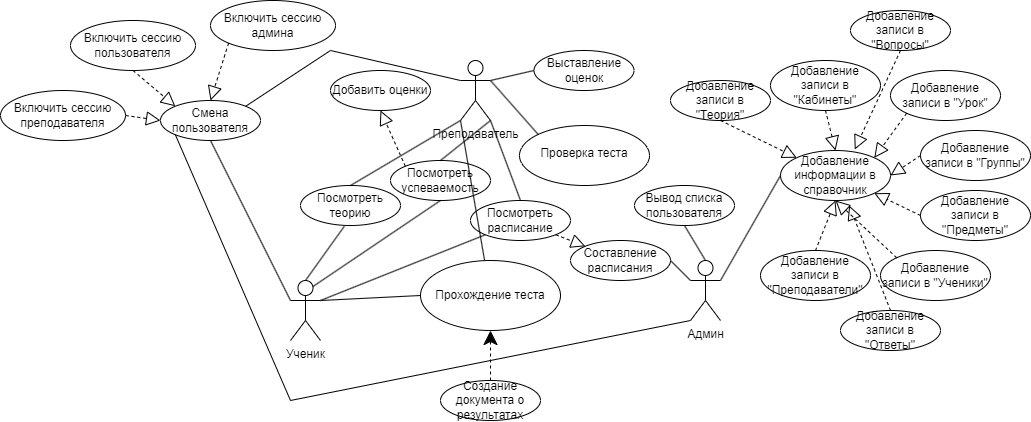
**СписокПользователей** –

1. Выводит список пользователей системы в меню админа

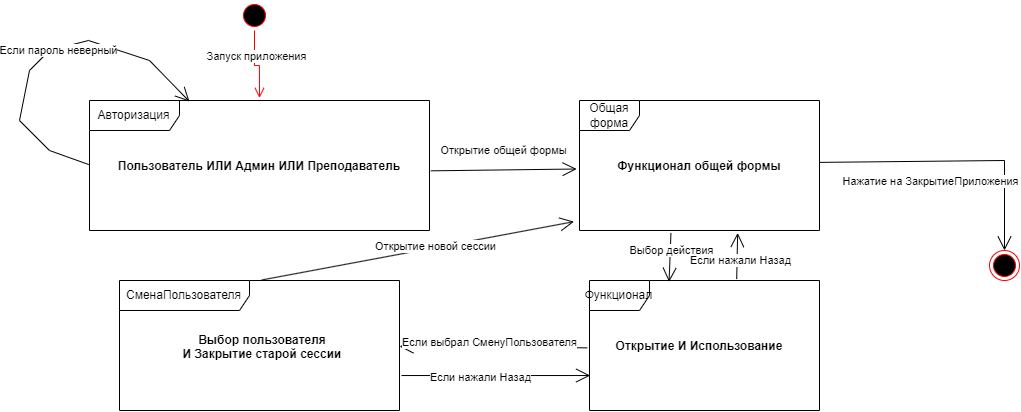
**Добавление объектов** –

1. Возможность добавления информации в справочники
   1. Диаграмма вариантов использования

Диаграмма вариантов использования-это графическое изображение возможных взаимодействий пользователя с системой. Диаграмма вариантов использования показывает различные варианты использования и различные типы пользователей, которые есть в системе, и часто будет сопровождаться другими типами диаграмм.

Диаграмма вариантов использования представляет собой граф специального вида, который является графической нотацией для представления конкретных вариантов использования, актеров, возможно некоторых интерфейсов, и отношений между этими элементами. Отдельные компоненты диаграммы могут быть заключены в прямоугольник, который обозначает проектируемую систему в целом. 

* 1. Диаграмма состояний

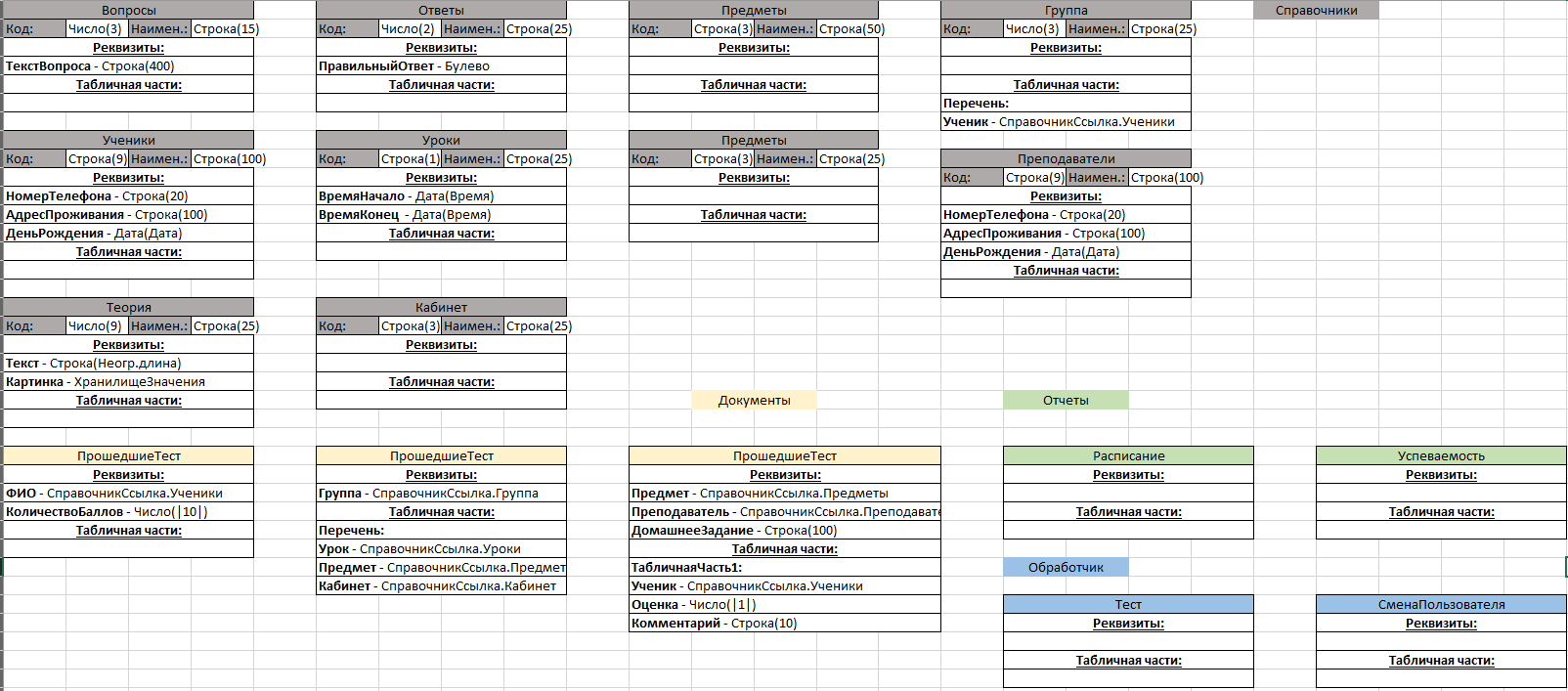
**Диаграмма состояний** - это тип диаграммы, используемый в информатике и смежных областях для описания поведения систем. Диаграммы состояний требуют, чтобы описываемая система состояла из конечного числа состояний; иногда это действительно так, в то время как в других случаях это разумная абстракция. Существует множество форм диаграмм состояний, которые немного отличаются и имеют разную семантику. 

* 1. Схема данных

**Схема** базы данных— её структура, описанная на формальном языке, поддерживаемом СУБД. В реляционных базах данных схема определяет таблицы, поля в каждой таблице (обычно с указанием их названия, типа, обязательности), и ограничения целостности (первичный, потенциальные и внешние ключи и другие ограничения).

Схемы в общем случае хранятся в словаре данных. Хотя схема определена на языке базы данных в виде текста, термин часто используется для обозначения графического представления структуры базы данных[[2]](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85#cite_note-2).

Основными объектами графического представления схемы являются таблицы и связи, определяемые внешними ключами.



* 1. Пользовательские сценарии

Test case №1:

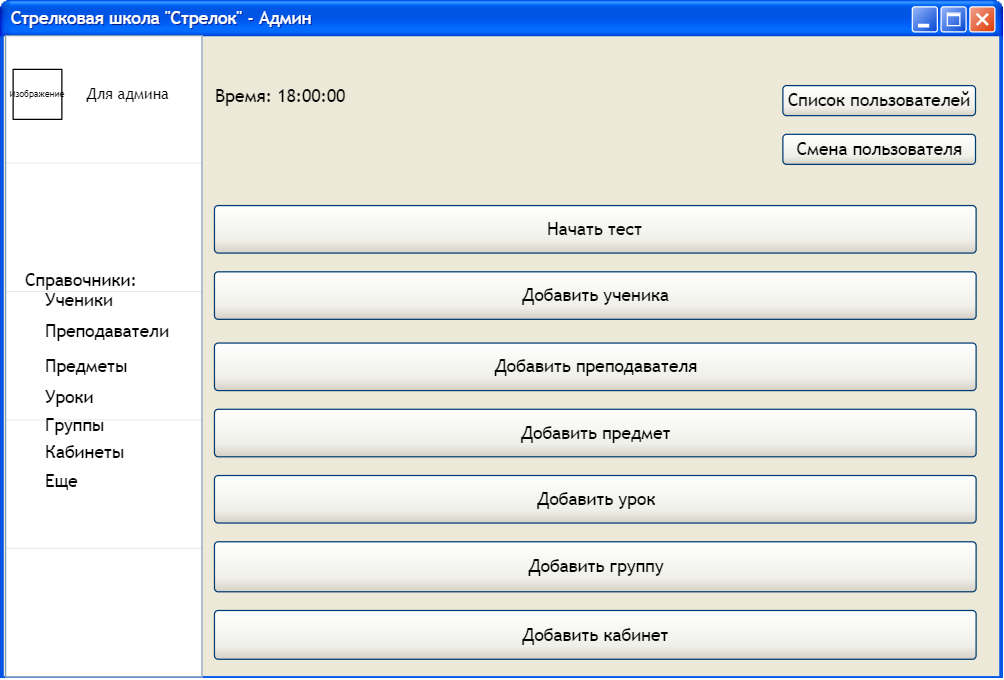
|  |  |
| --- | --- |
| **Test Case №** | ТС\_1 |
| **Приоритет теста** | Высокий |
| **Название тестирования/Имя** | Авторизация пользователя в систему. |
| **Резюме испытания** | Пользователь должен авторизоваться в систему при правильном логине, так как в учебной версии недоступно установления пароля. |
| **Шаги тестирования** | 1. Запуск программы. 2. Ввод логина в поле для логина. 3. Нажатие на кнопку «Войти» |
| **Ожидаемый результат** | Пользователь,админ или преподаватель авторизуется в систему. |
| **Фактический результат** | Авторизация прошла успешно. |
| **Предпосылки** | Только поле будет пароля будет пустовать. |
| **Статус (Зачет/Незачет)** | Зачет |
| **Комментарии** | 1. Протестирована авторизация с неправильными данными. В результате программа выводит оповещение о не пройденной идентификации. 2. Пропускает только при совпадении поля логина с соответствующими в базе. |

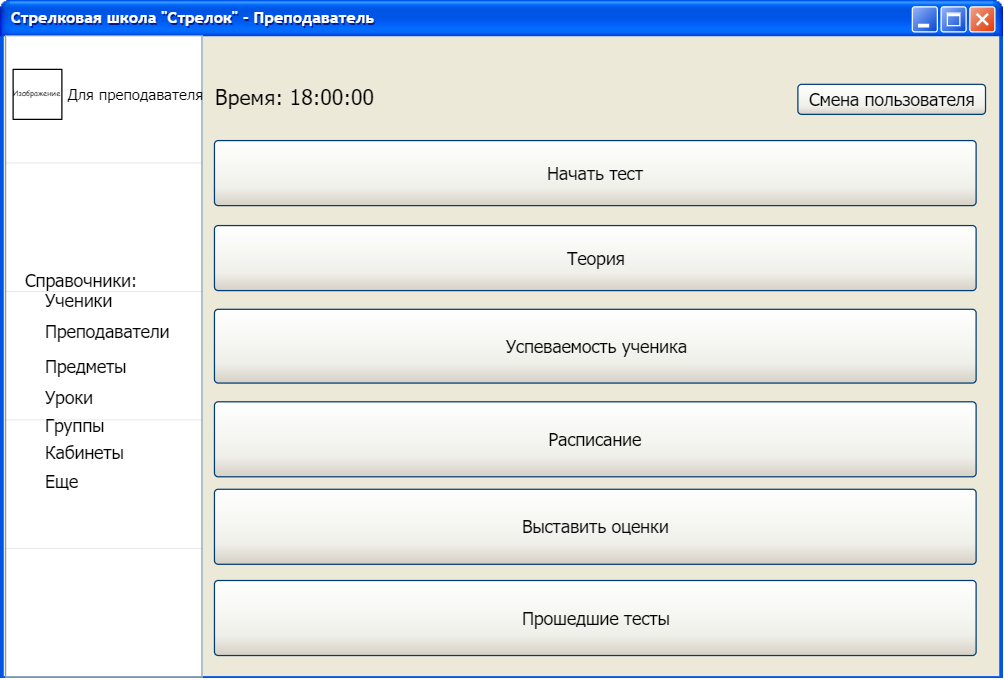
Test case №2:

|  |  |
| --- | --- |
| **Test Case №** | ТС\_2 |
| **Приоритет теста** | Среднее |
| **Название тестирования/Имя** | Загрузка фотографий в справочник «Теория». |
| **Резюме испытания** | Преподаватель загружает фотографию, справочник должен эту фотографию загрузить. |
| **Шаги тестирования** | 1. Запуск программы под пользователем «Преподаватель». 2. Нажимаем кнопку «Теория», после того, как выбрали запись в справочнике, нажимаем на картинку, чтобы попробовать загрузить. 3. Загружаем 2 разные расширением файла фотографии. |
| **Ожидаемый результат** | Две этих фотографий будут подходить для загрузки в справочник. |
| **Фактический результат** | В окне выбора видно лишь фотографию с расширением .jpg |
| **Предпосылки** |  |
| **Статус (Зачет/Незачет)** | Незачет |
| **Комментарии** | 1. Для тестирования были взяты 2 фотографии, одна в .jpg расширении, другая в .png, но загрузить получилось в .jpg |

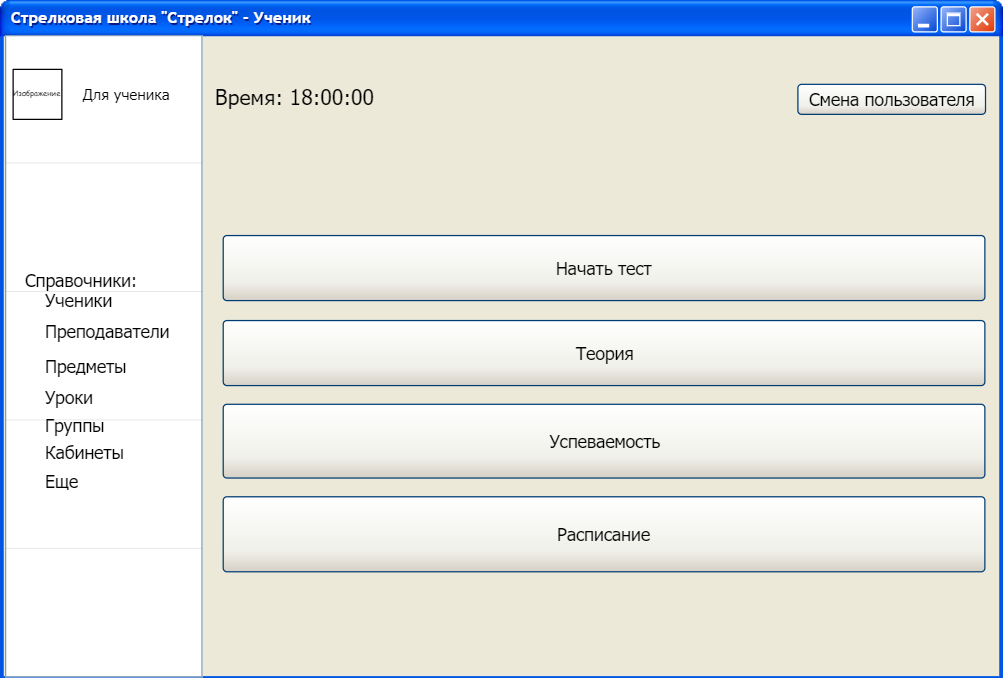
* 1. Прототипы основных интерфейсов

**Прототип системы** — макет (черновой, пробной версии) программы, обычно — с целью проверки пригодности предлагаемых для применения концепций, архитектурных и/или технологических решений, а также для представления программы заказчику на ранних стадиях процесса разработки. В ходе разработки интерфейсов было много идей для вида окон, но в ходе созданий прототипов убирались лишние идеи и придерживался минимализма, чтобы систему лишними декорациями не нагружать. В ходе разработки интерфейса некоторые функции, представленные на прототипах не были добавлены, так как некоторые не были реализованы.

  
Прототип окна админа



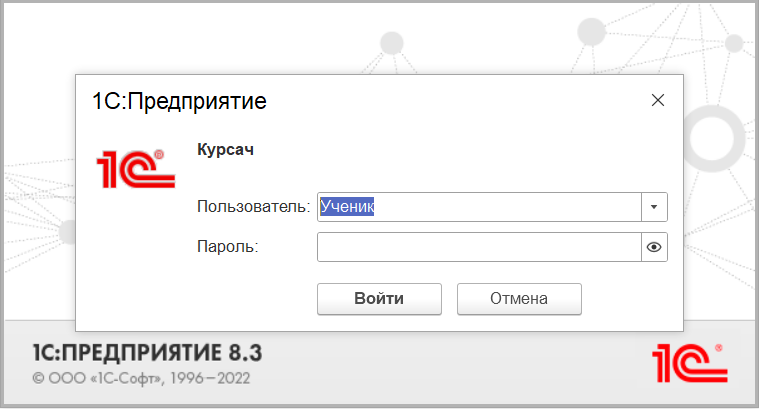
Прототип окна преподавателя

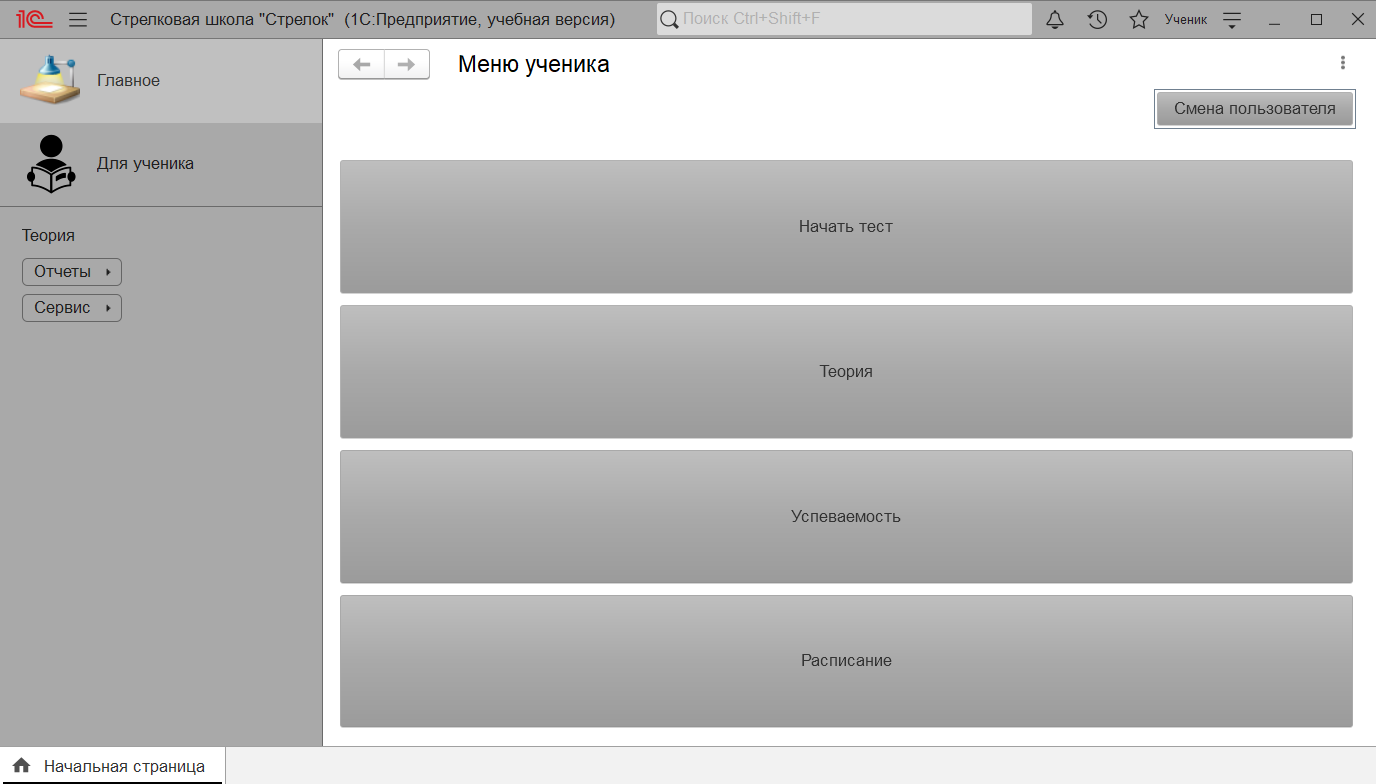


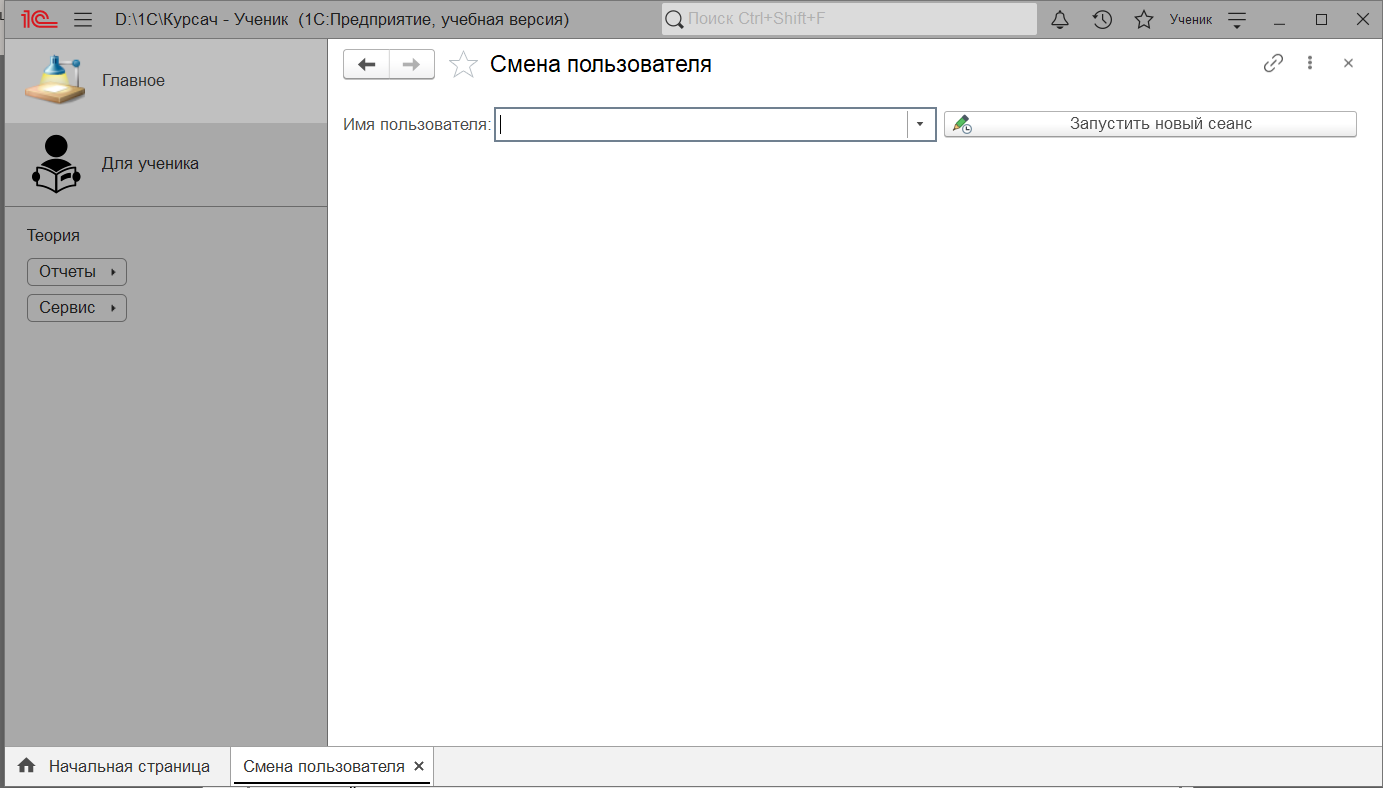
Прототип окна ученика

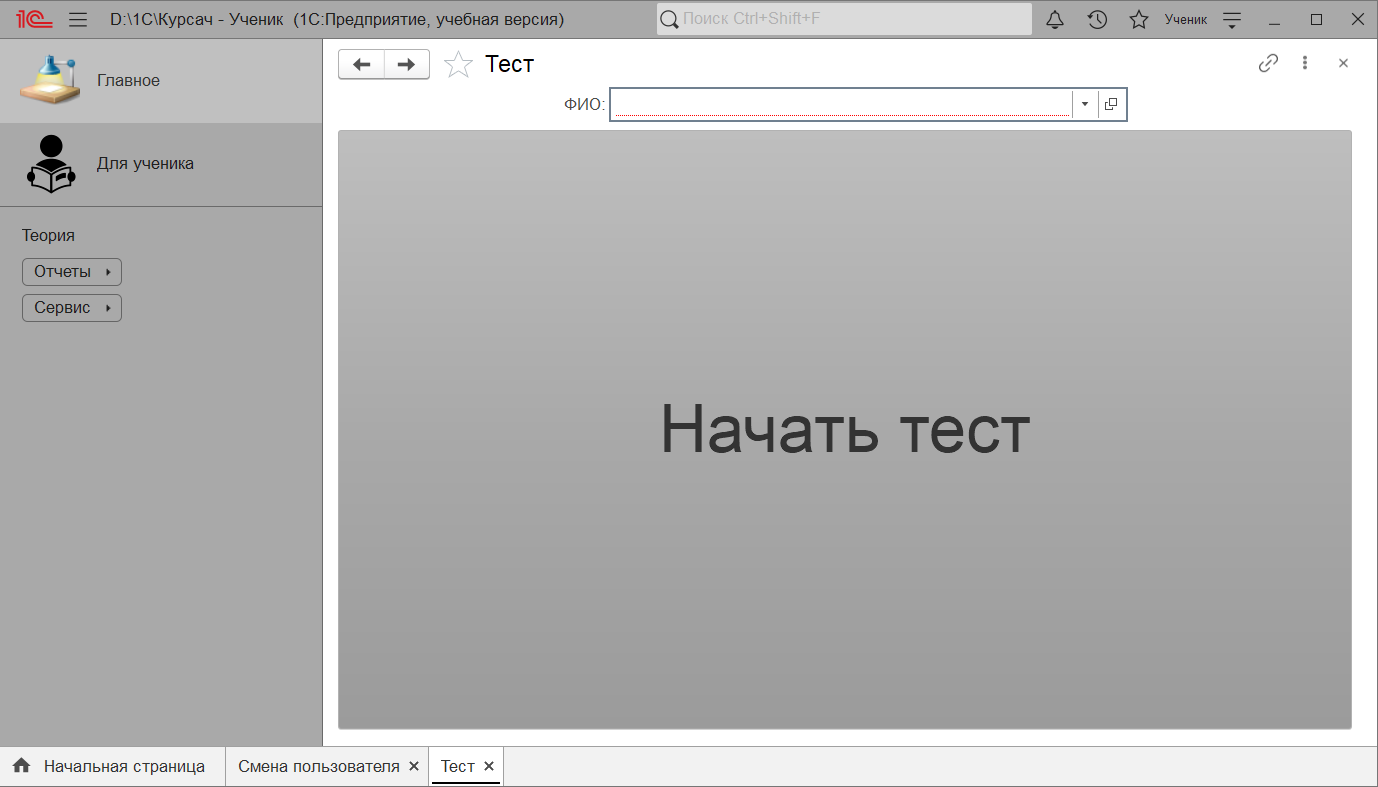
ЗАКЛЮЧЕНИЕ

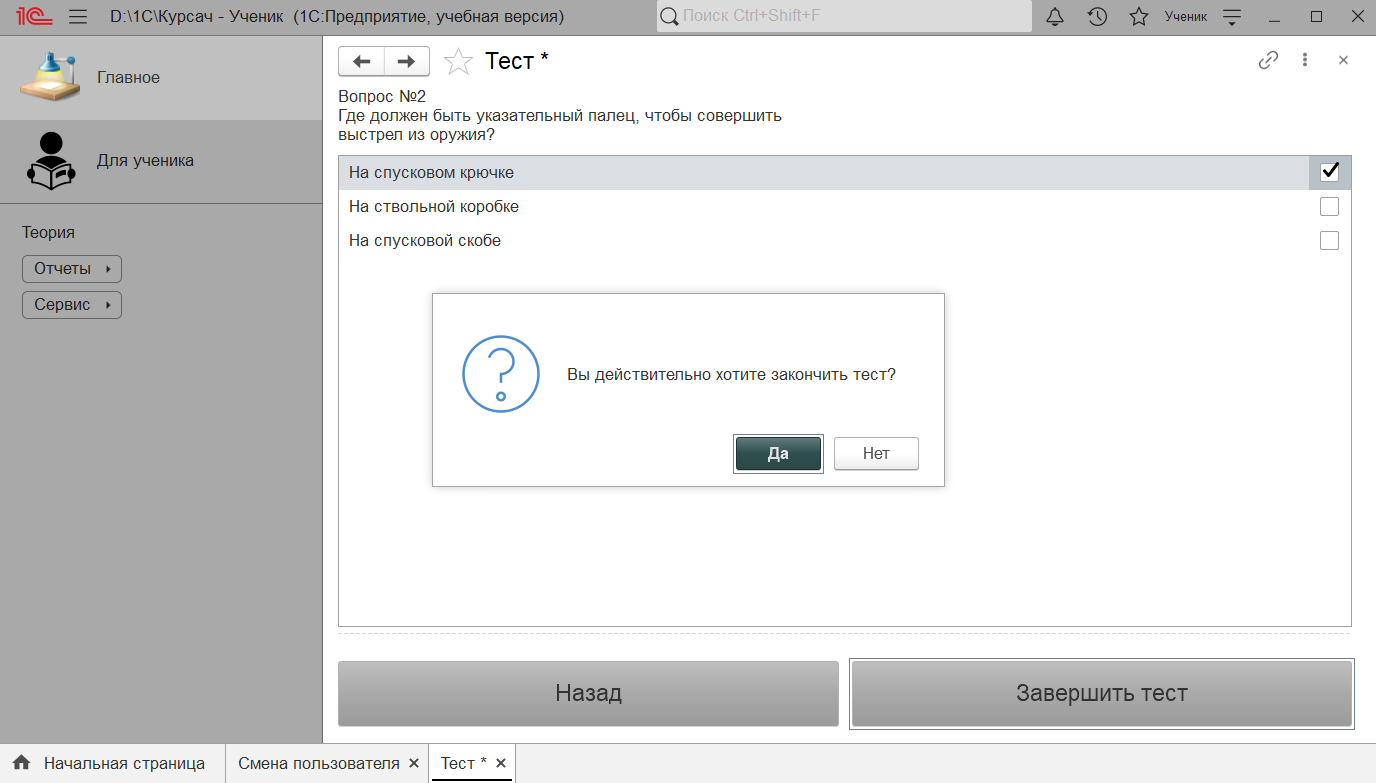
* 1. Приложение A. Листинг окон программы

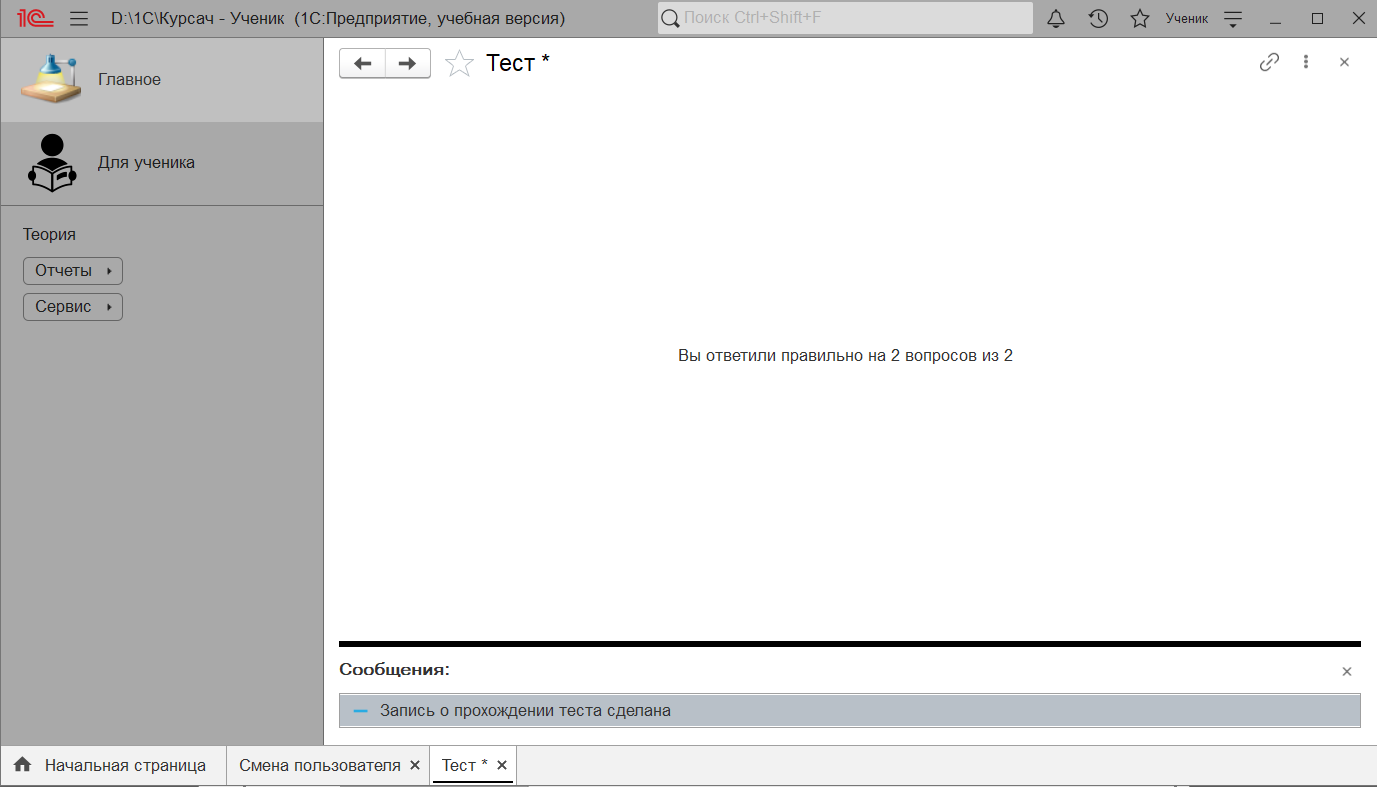


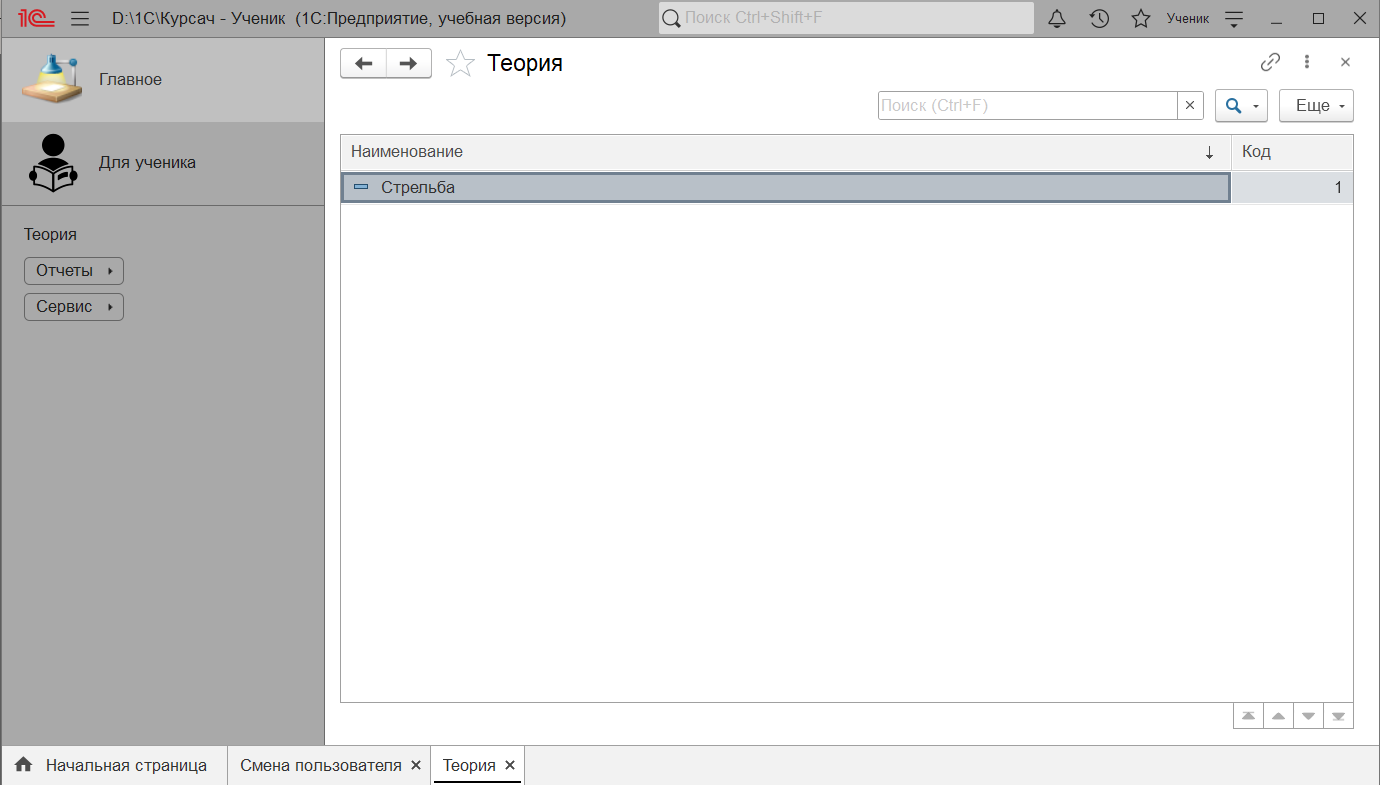


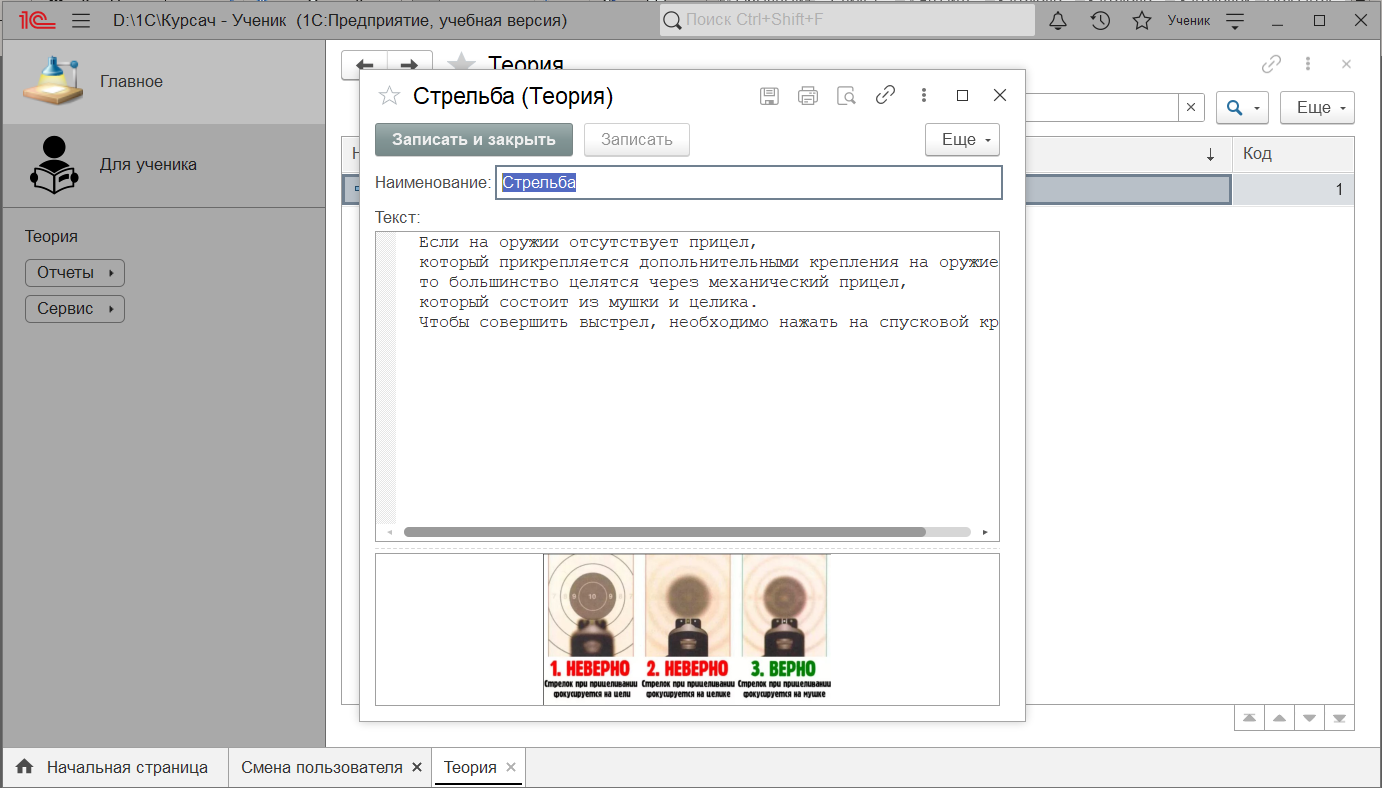


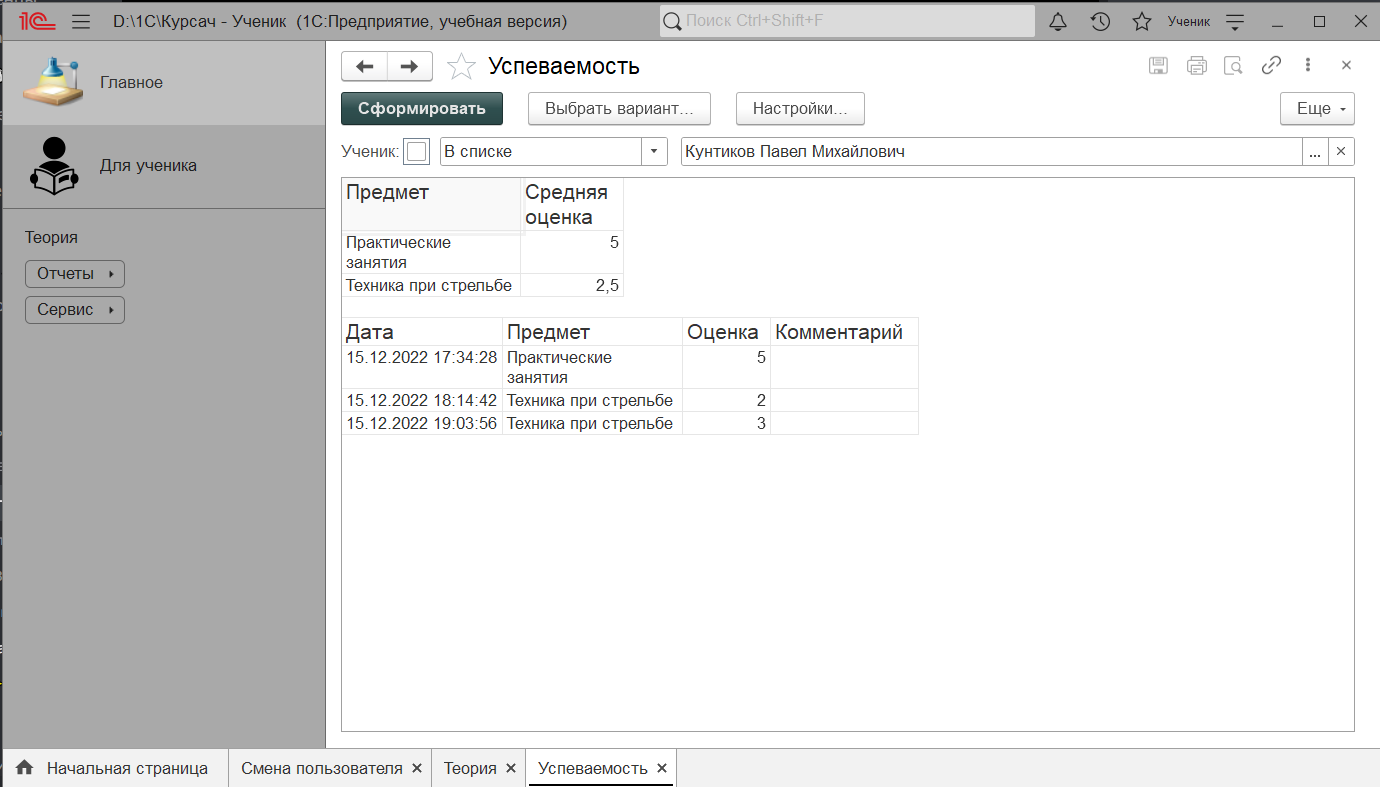


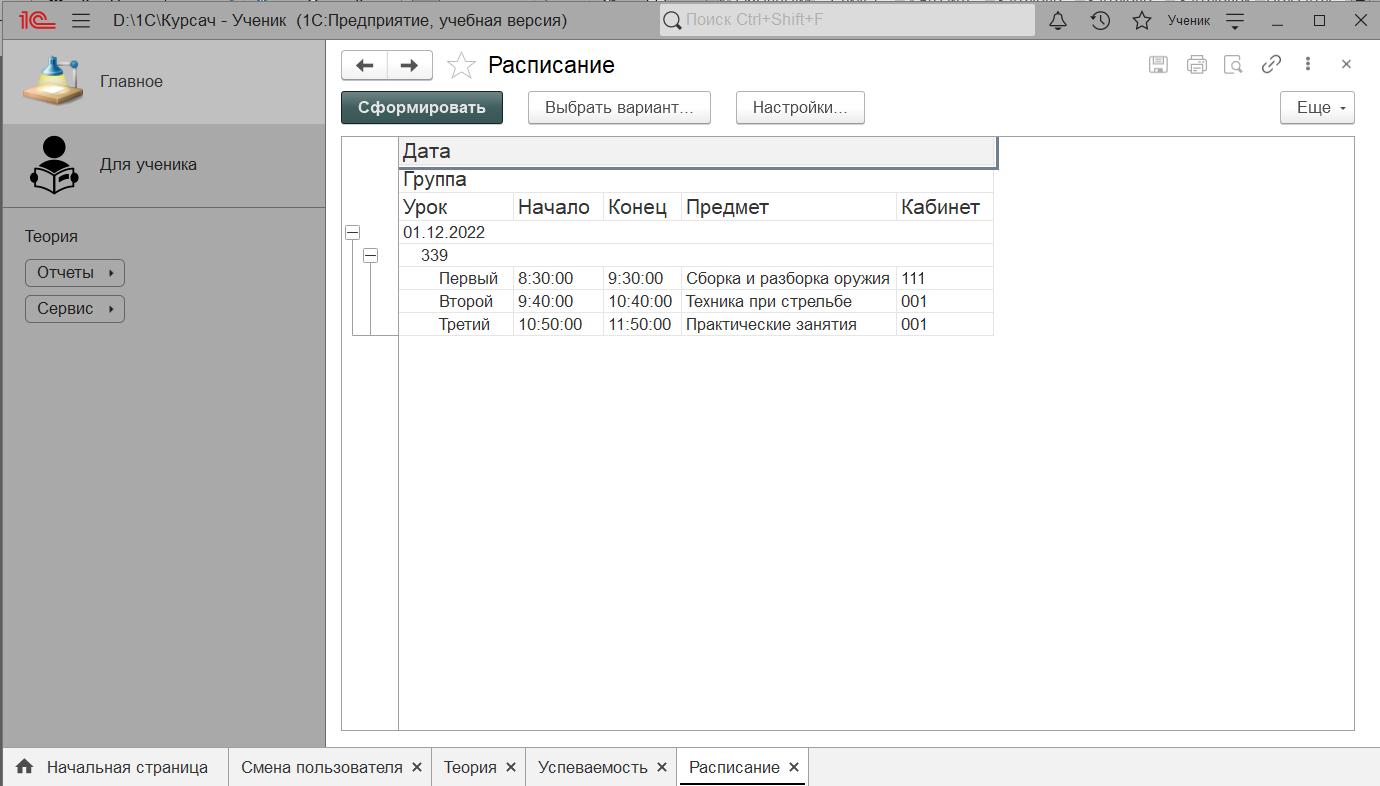


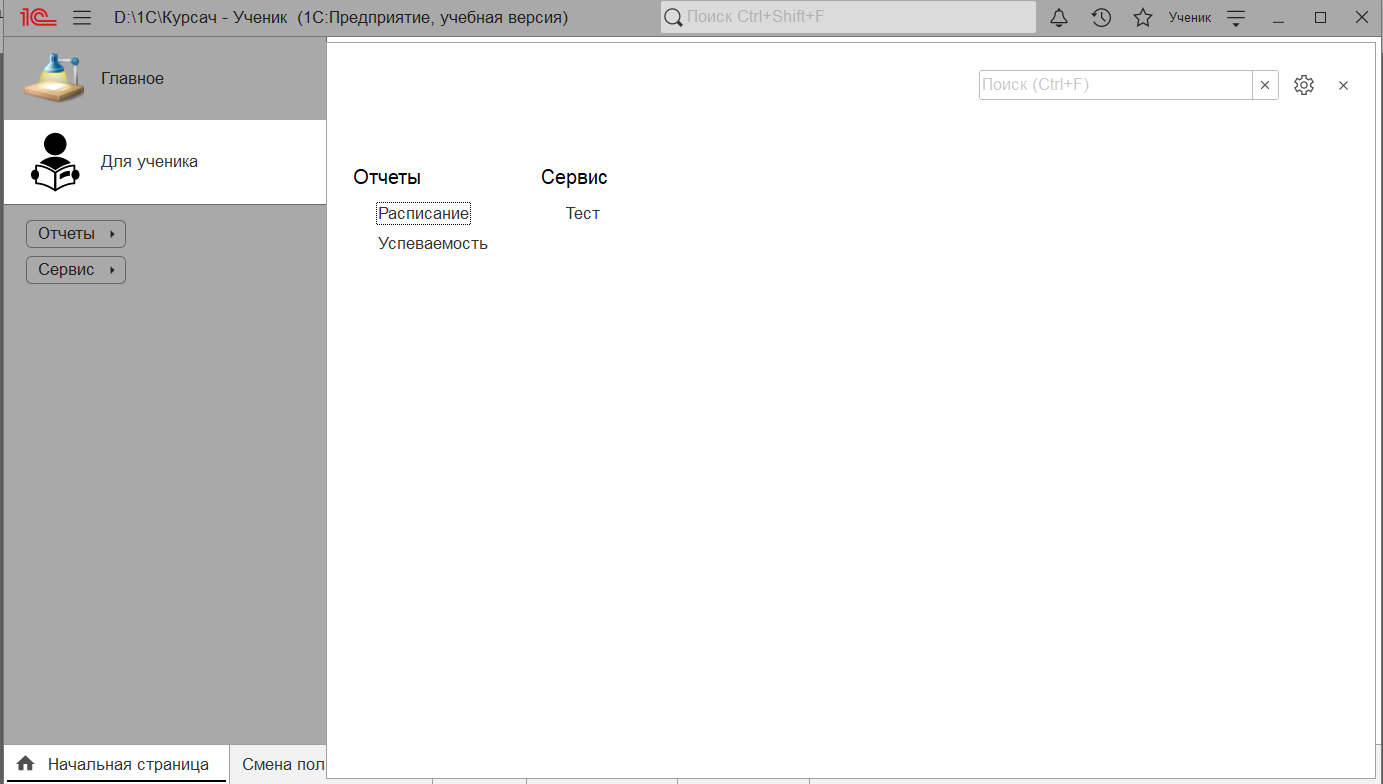


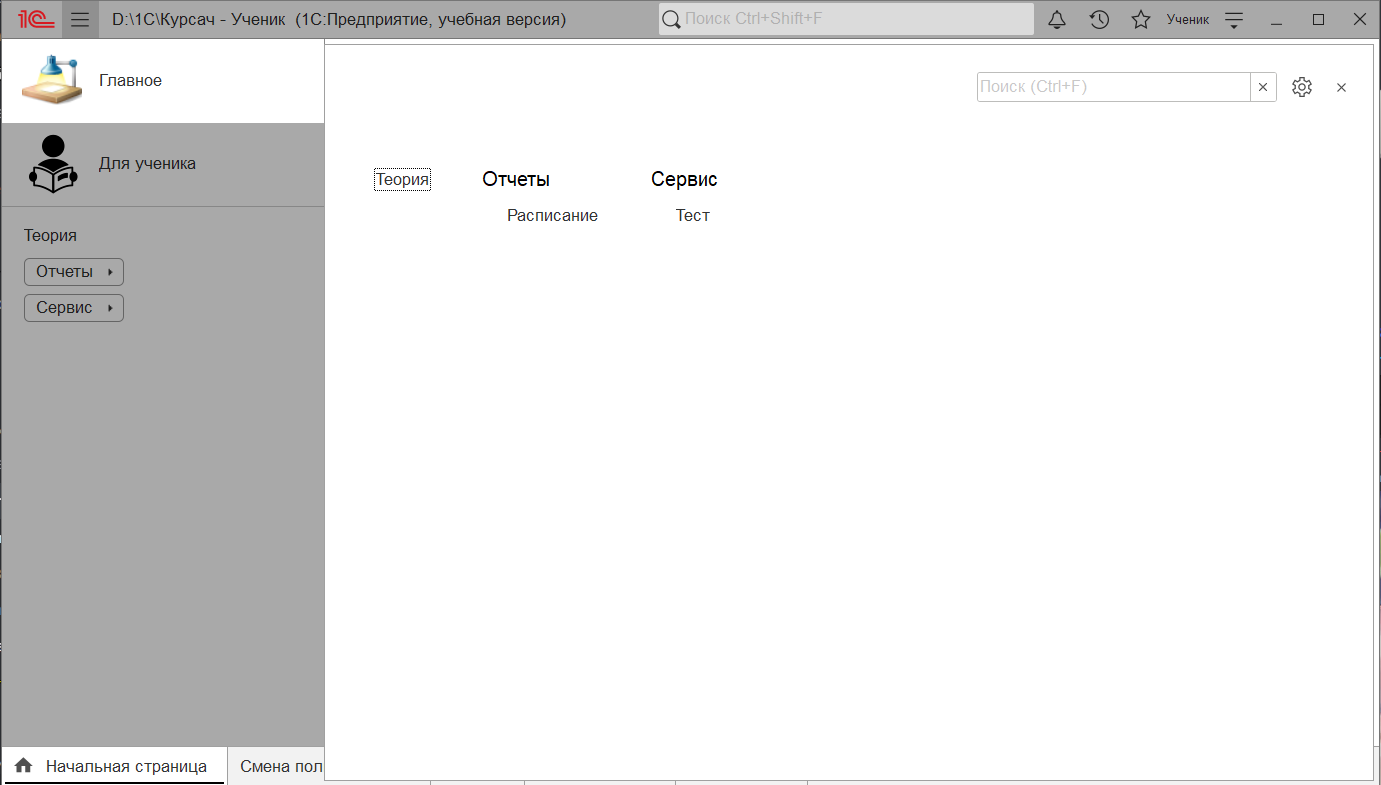


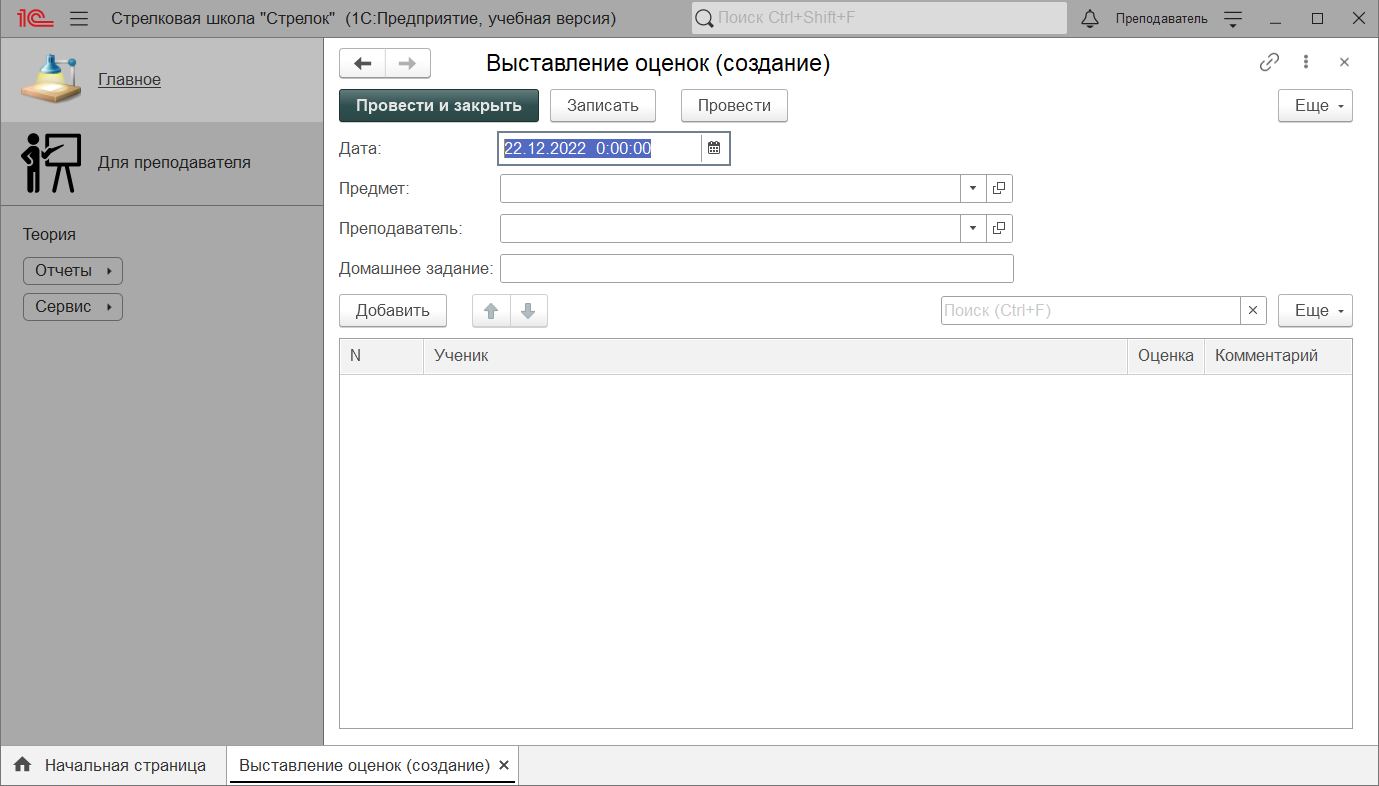


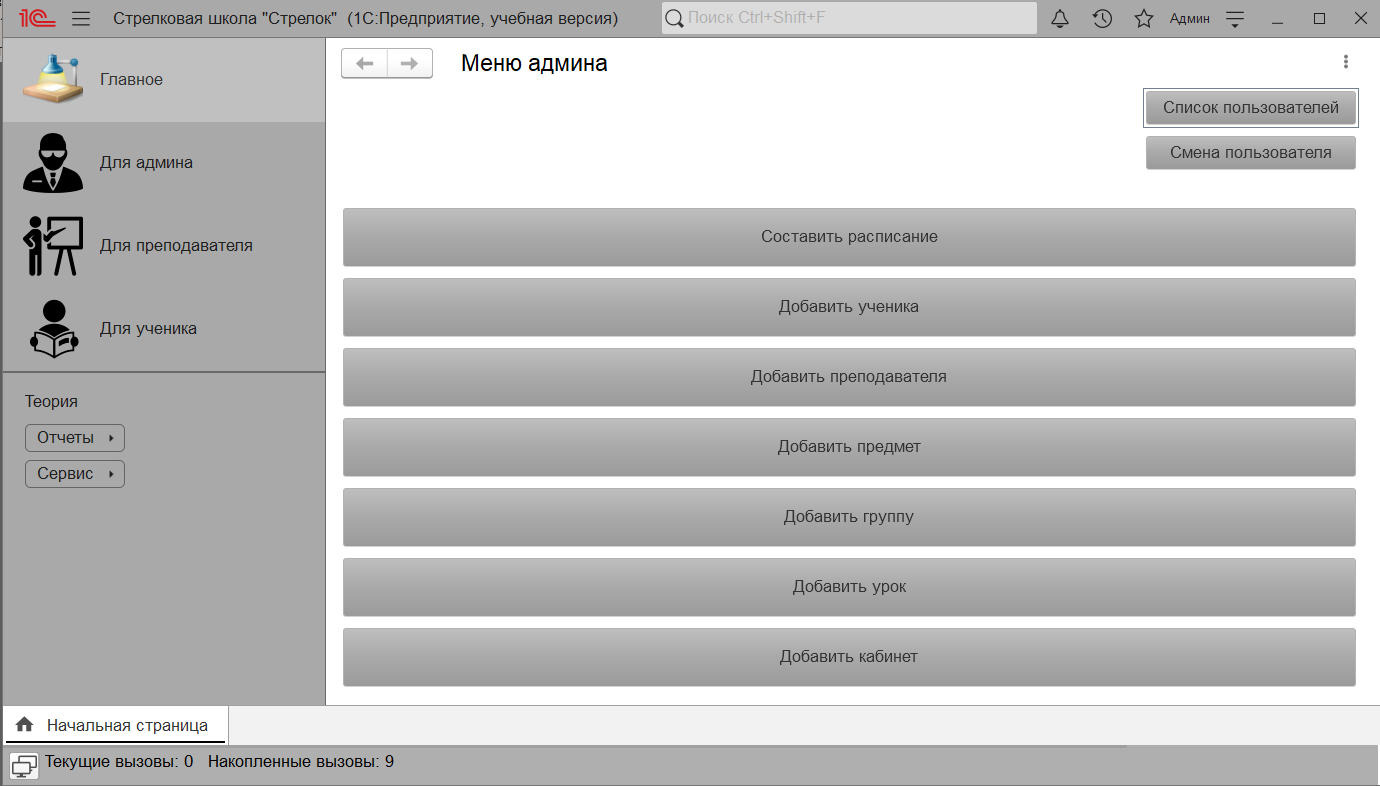


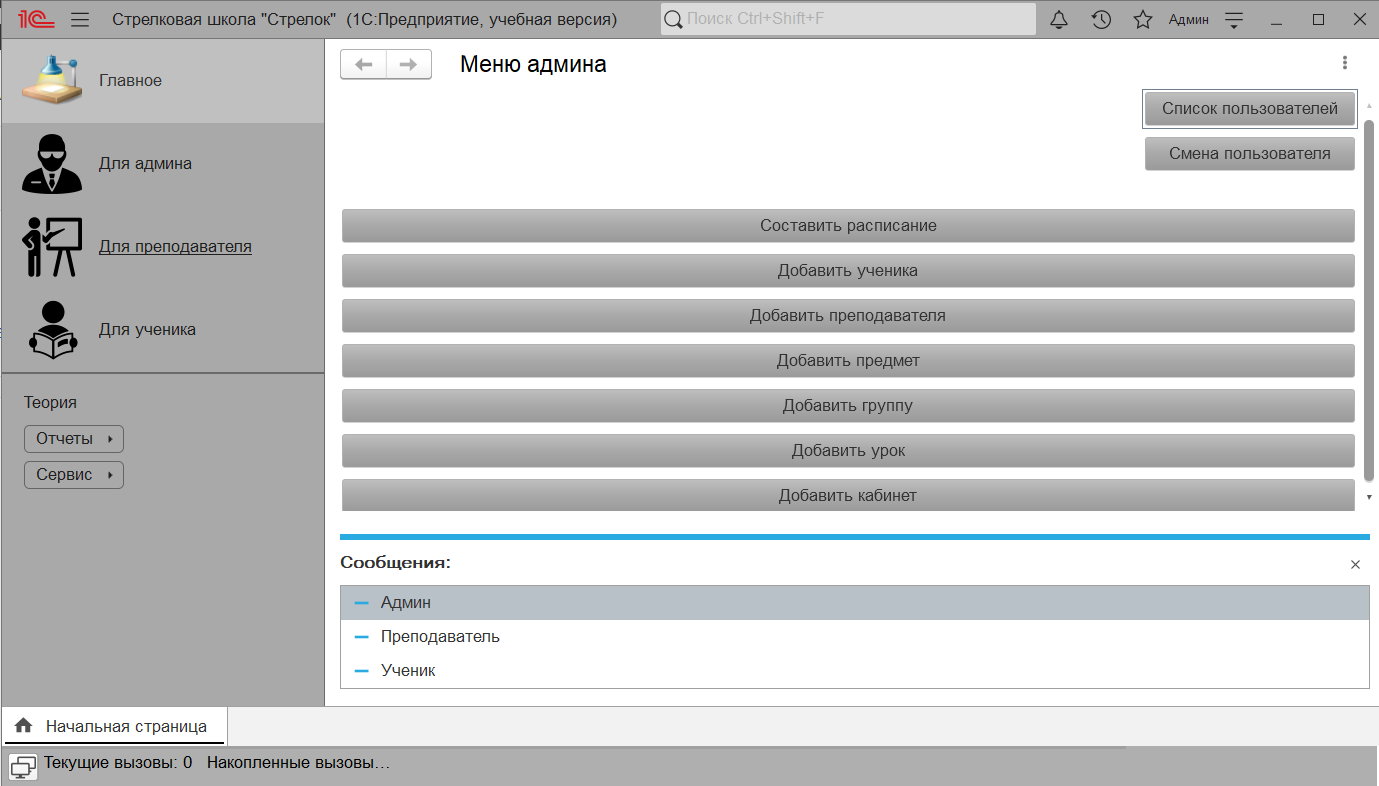


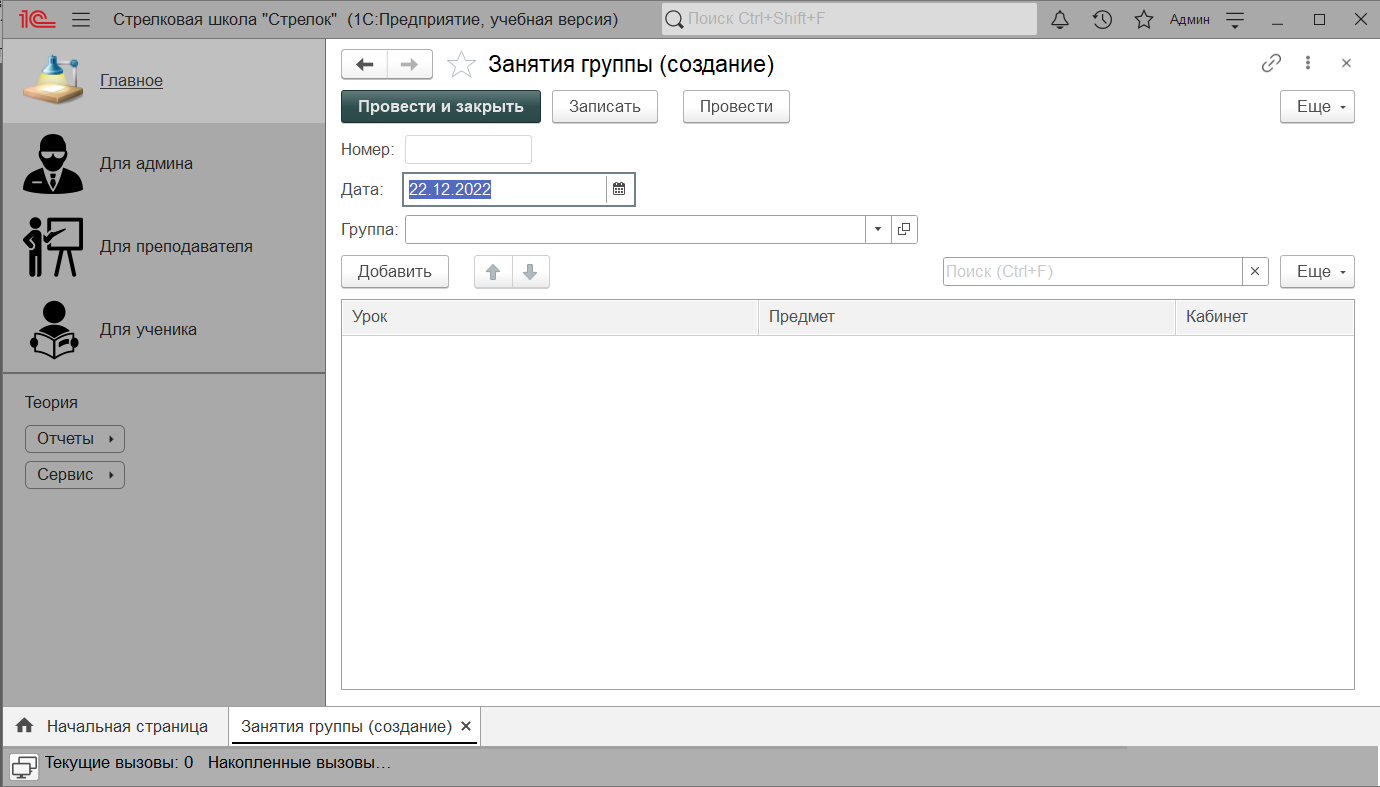


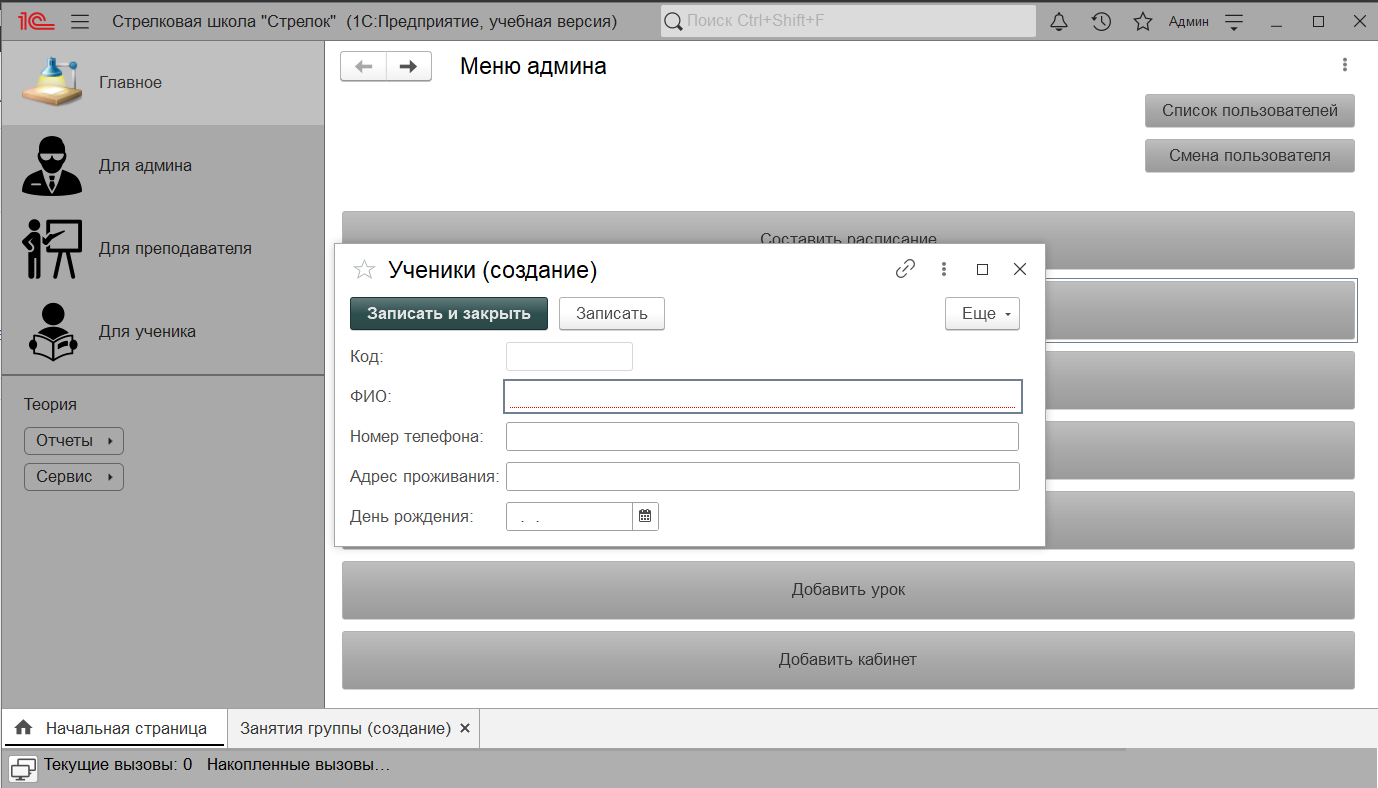


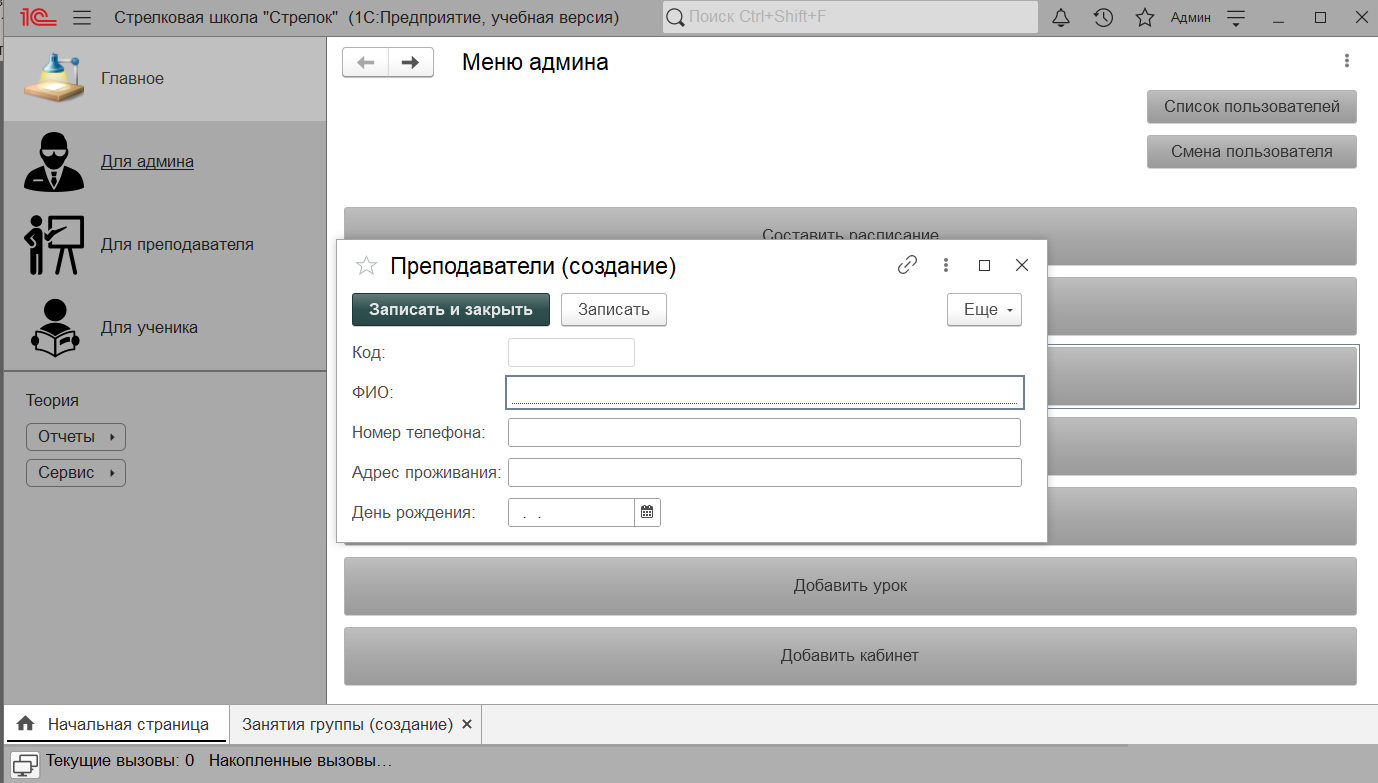


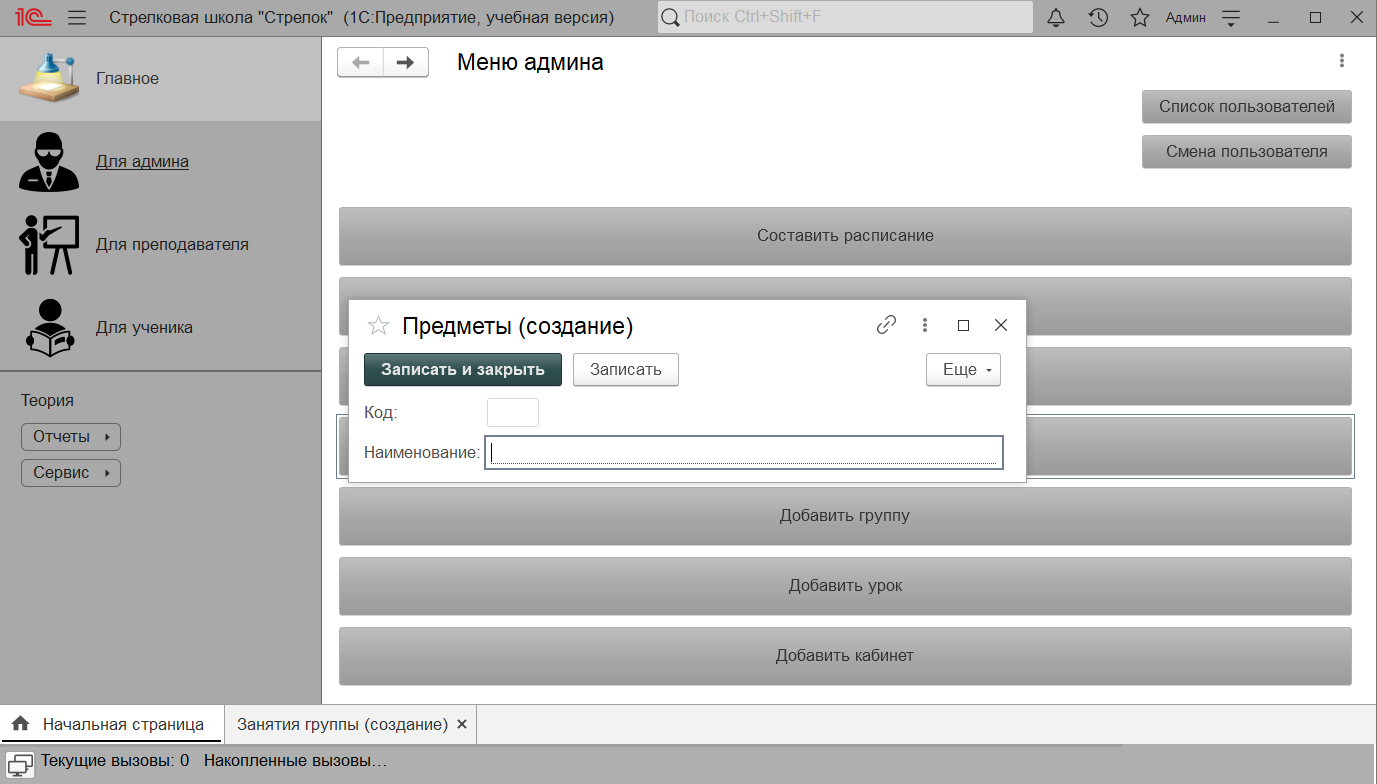


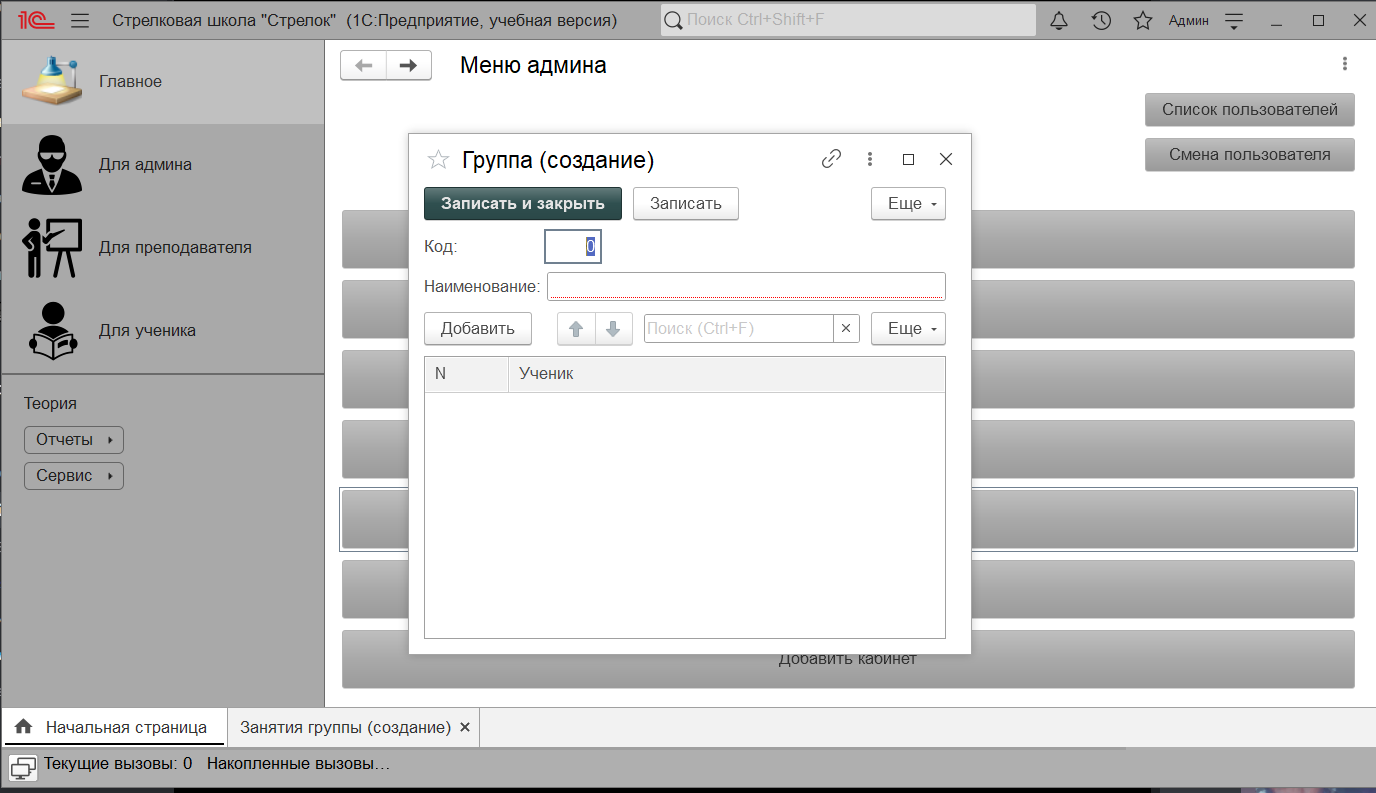


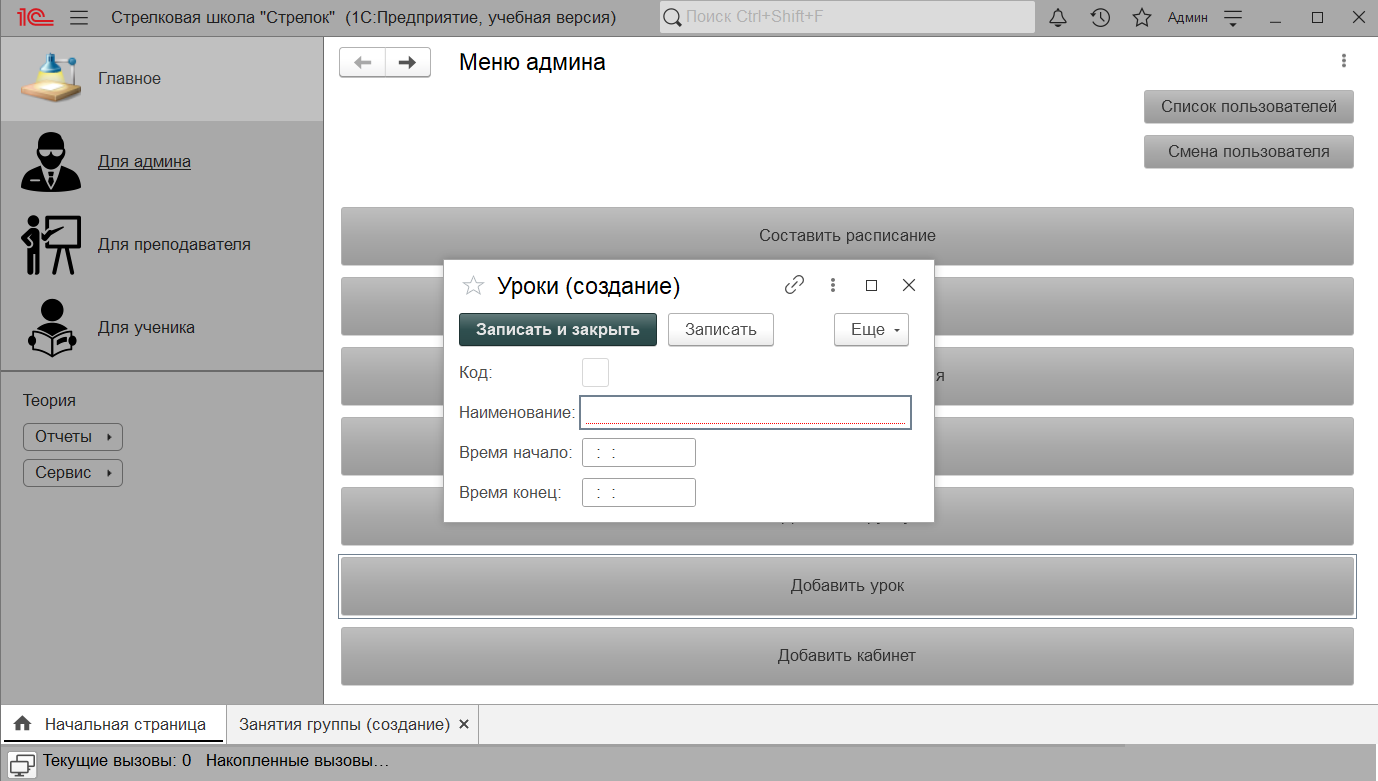


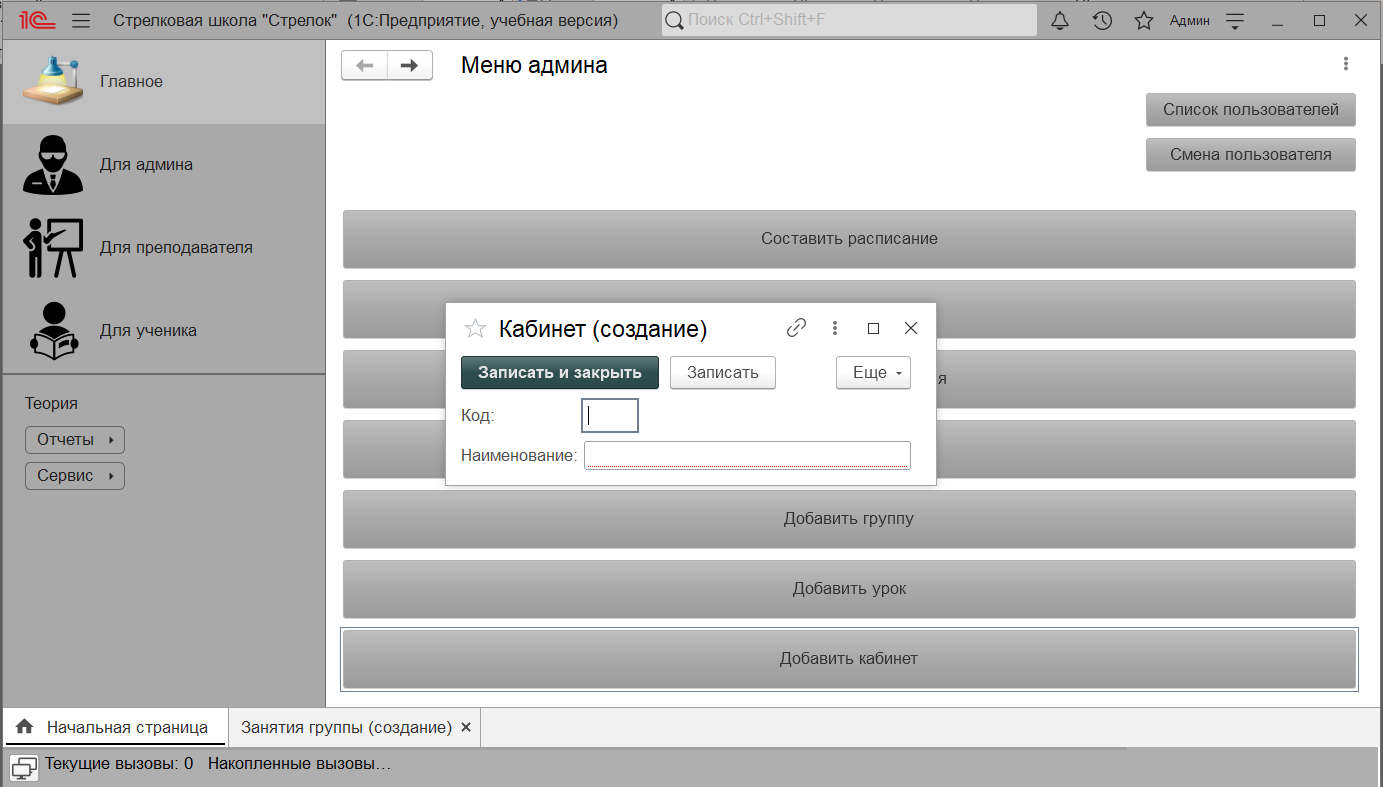


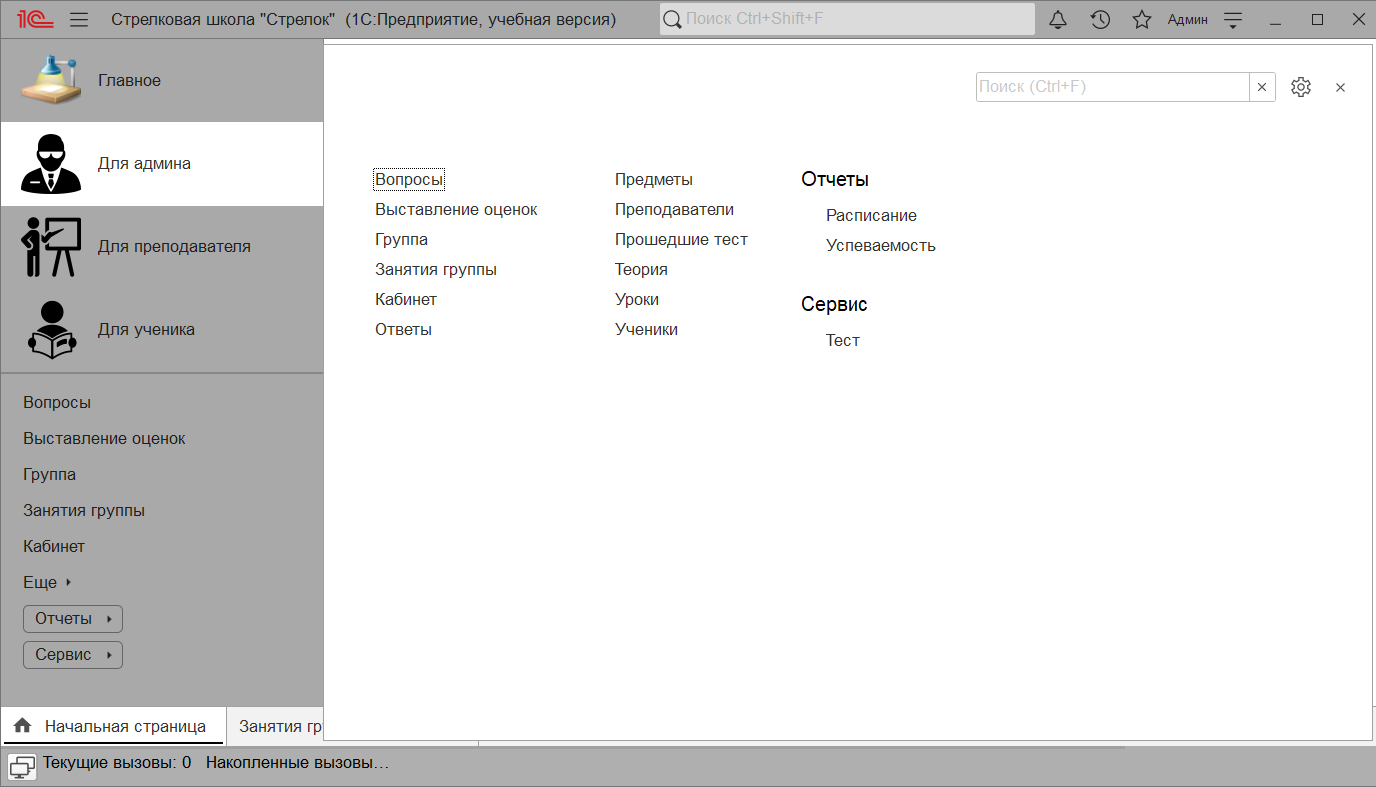












* 1. Приложение В. Листинг кода программы
     1. Модуль общей формы МенюПреподавателя:

&НаКлиенте

Процедура Расписание(Команда)

ОткрытьФорму("Отчет.Расписание.Форма.ФормаОтчета");

КонецПроцедуры

&НаКлиенте

Процедура НачатьТест(Команда)

ОткрытьФорму("Обработка.Тест.Форма.Форма");

КонецПроцедуры

&НаКлиенте

Процедура ВыставитьОценки(Команда)

ОткрытьФорму("Документ.ВыставлениеОценок.Форма.ФормаДокумента");

КонецПроцедуры

&НаКлиенте

Процедура УспеваемостьУченика(Команда)

ОткрытьФорму("Отчет.Успеваемость.Форма.ФормаОтчета");

КонецПроцедуры

&НаКлиенте

Процедура Теория(Команда)

ОткрытьФорму("Справочник.Теория.Форма.ФормаСписка");

КонецПроцедуры

&НаКлиенте

Процедура СменаПользователя(Команда)

ОткрытьФорму("Обработка.СменаПользователя.Форма.MainForm");

КонецПроцедуры

&НаКлиенте

Процедура ПрошедшиеТесты(Команда)

ОткрытьФорму("Документ.ПрошедшиеТест.Форма.ФормаСписка");

КонецПроцедуры

* + 1. Модуль общей формы МенюУченика:

&НаКлиенте

Процедура НачатьТест(Команда)

ОткрытьФорму("Обработка.Тест.Форма.Форма");

КонецПроцедуры

&НаКлиенте

Процедура Расписание(Команда)

ОткрытьФорму("Отчет.Расписание.Форма.ФормаОтчета");

КонецПроцедуры

&НаКлиенте

Процедура Успеваемость(Команда)

ОткрытьФорму("Отчет.Успеваемость.Форма.ФормаОтчета");

КонецПроцедуры

&НаКлиенте

Процедура СменаПользователя(Команда)

ОткрытьФорму("Обработка.СменаПользователя.Форма.MainForm");

КонецПроцедуры

&НаКлиенте

Процедура Теория(Команда)

ОткрытьФорму("Справочник.Теория.Форма.ФормаСписка");

КонецПроцедуры

* + 1. Модуль общей формы МенюАдмина

&НаКлиенте

Процедура СписокПользователей(Команда)

ВывестиСписокПользователей();

КонецПроцедуры

&НаСервере

Процедура ВывестиСписокПользователей() Экспорт

СписокПользователей = ПользователиИнформационнойБазы.ПолучитьПользователей();

Для Каждого ПользовательИБ Из СписокПользователей Цикл

Сообщить(ПользовательИБ);

КонецЦикла;

КонецПроцедуры

&НаКлиенте

Процедура СменаПользователя(Команда)

ОткрытьФорму("Обработка.СменаПользователя.Форма.MainForm");

КонецПроцедуры

&НаКлиенте

Процедура СоставитьРасписание(Команда)

ОткрытьФорму("Документ.ЗанятияГруппы.Форма.ФормаДокумента");

КонецПроцедуры

&НаКлиенте

Процедура ДобавитьУченика(Команда)

ОткрытьФорму("Справочник.Ученики.Форма.ФормаЭлемента");

КонецПроцедуры

&НаКлиенте

Процедура ДобавитьПредмет(Команда)

ОткрытьФорму("Справочник.Предметы.Форма.ФормаЭлемента");

КонецПроцедуры

&НаКлиенте

Процедура ДобавитьГруппу(Команда)

ОткрытьФорму("Справочник.Группа.Форма.ФормаЭлемента");

КонецПроцедуры

&НаКлиенте

Процедура ДобавитьУрок(Команда)

ОткрытьФорму("Справочник.Уроки.Форма.ФормаЭлемента");

КонецПроцедуры

&НаКлиенте

Процедура ДобавитьКабинет(Команда)

ОткрытьФорму("Справочник.Кабинет.Форма.ФормаЭлемента");

КонецПроцедуры

&НаКлиенте

Процедура ДобавитьПреподавателя(Команда)

ОткрытьФорму("Справочник.Преподаватели.Форма.ФормаЭлемента");

КонецПроцедуры

* + 1. Модуль формы элемента справочника Теория

&НаКлиенте

Процедура СсылкаНаКартинкуНажатие(Элемент, СтандартнаяОбработка)

СтандартнаяОбработка = Ложь;

Если (ИмяПользователя() = "Преподаватель") ИЛИ (ИмяПользователя() = "Админ") Тогда

Режим = РежимДиалогаВыбораФайла.Открытие;

ДиалогОткрытия = Новый ДиалогВыбораФайла(Режим);

ДиалогОткрытия.ПолноеИмяФайла = "";

Фильтр = "Файл Jpg (\*.jpg)|\*.jpg";

ДиалогОткрытия.Фильтр = Фильтр;

ДиалогОткрытия.МножественныйВыбор = Ложь;

ДиалогОткрытия.Заголовок = "Выберете файл для загрузки";

ОписаниеОповещения = Новый ОписаниеОповещения("ПослеЗагрузкиФайла",ЭтаФорма);

ДиалогОткрытия.Показать(ОписаниеОповещения);

КонецЕсли;

КонецПроцедуры

&НаКлиенте

Процедура ПослеЗагрузкиФайла(ВыбранныйФайл,ДопПараметр) Экспорт

Если ВыбранныйФайл = Неопределено Тогда

Возврат;

КонецЕсли;

ОписаниеОповещения = Новый ОписаниеОповещения("ПослеПомещенияФайла", ЭтаФорма);

НачатьПомещениеФайла(ОписаниеОповещения,, ВыбранныйФайл[0], Ложь, УникальныйИдентификатор);

КонецПроцедуры

&НаКлиенте

Процедура ПослеПомещенияФайла(Результат, Адрес, ВыбранноеИмяФайла,ДопПараметры) Экспорт

Если Не Результат Тогда

Возврат;

КонецЕсли;

СсылкаНаКартинку = Адрес;

Модифицированность = Истина;

КонецПроцедуры

&НаСервере

Процедура ПередЗаписьюНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)

Если ЭтоАдресВременногоХранилища(СсылкаНаКартинку) Тогда

ФайлКартинки = ПолучитьИзВременногоХранилища(СсылкаНаКартинку);

ТекущийОбъект.Картинка = Новый ХранилищеЗначения(ФайлКартинки);

УдалитьИзВременногоХранилища(СсылкаНаКартинку);

СсылкаНаКартинку = ПолучитьНавигационнуюСсылку(Объект.Ссылка,"Картинка");

КонецЕсли;

КонецПроцедуры

&НаСервере

Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

СсылкаНаКартинку = ПолучитьНавигационнуюСсылку(Объект.Ссылка,"Картинка");

КонецПроцедуры

* + 1. Запрос схемы компоновки данных отчета Расписание

ВЫБРАТЬ

ЗанятияГруппы.Дата КАК Дата,

ЗанятияГруппы.Группа КАК Группа,

ЗанятияГруппы.Перечень.(

Урок КАК Урок,

Предмет КАК Предмет,

Кабинет КАК Кабинет,

Урок.ВремяНачало КАК УрокВремяНачало,

Урок.ВремяКонец КАК УрокВремяКонец

) КАК Перечень

ИЗ

Документ.ЗанятияГруппы КАК ЗанятияГруппы

* + 1. Запрос схемы компоновки данных отчета Успеваемость

ВЫБРАТЬ

ВыставлениеОценок.Дата КАК Дата,

ВыставлениеОценок.Предмет КАК Предмет,

ВыставлениеОценок.ТабличнаяЧасть1.(

Ученик КАК Ученик,

Оценка КАК Оценка,

Комментарий КАК Комментарий

) КАК ТабличнаяЧасть1

ИЗ

Документ.ВыставлениеОценок КАК ВыставлениеОценок

* + 1. Модуль формы обработки Тест

&НаКлиенте

Процедура НачатьТест(Команда)

ТестВПроцессе = Истина;

Если ПустаяСтрока(ФИО) Тогда

Отказ = Истина;

Сообщить("Не заполнено ваше ФИО");

Возврат;

КонецЕсли;

ЗапВопросыОтветы();

Если ВопросыОтветы.Количество() = 0 Тогда

Сообщить("Тест не найден. Обратитесь к администратору");

Возврат;

КонецЕсли;

Элементы.Начало.Видимость = Ложь;

Элементы.Тест.Видимость = Истина;

НомерТекущегоВопроса = 1;

ЗапОтветы();

КонецПроцедуры

&НаКлиенте

Процедура ЗапОтветы()

УстановитьВидимость();

ЗапОтветыНаСервере();

КонецПроцедуры

&НаСервере

Процедура ЗапВопросыОтветы()

З = Новый Запрос;

З.Текст = "

|ВЫБРАТЬ

|Ссылка КАК Вопрос

|ИЗ Справочник.Вопросы

|ГДЕ НЕ ПометкаУдаления

|УПОРЯДОЧИТЬ ПО Код";

ВопросыОтветы.Загрузить(З.Выполнить().Выгрузить());

КонецПроцедуры

&НаКлиенте

Процедура УстановитьВидимость()

Если НомерТекущегоВопроса = 1 И НомерТекущегоВопроса = ВопросыОтветы.Количество() Тогда

Элементы.Назад.Видимость = Ложь;

Элементы.ЗавершитьТест.Видимость = Истина;

Элементы.Вперед.Видимость = Ложь;

ИначеЕсли НомерТекущегоВопроса = ВопросыОтветы.Количество() Тогда

Элементы.Вперед.Видимость = Ложь;

Элементы.ЗавершитьТест.Видимость = Истина;

Элементы.Назад.Видимость = Истина;

ИначеЕсли НомерТекущегоВопроса = 1 Тогда

Элементы.Вперед.Видимость = Истина;

Элементы.ЗавершитьТест.Видимость = Ложь;

Элементы.Назад.Видимость = Ложь;

Иначе

Элементы.Назад.Видимость = Истина;

Элементы.Вперед.Видимость = Истина;

Элементы.ЗавершитьТест.Видимость = Ложь;

КонецЕсли;

КонецПроцедуры

&НаСервере

Процедура ЗапОтветыНаСервере()

ВариантОтвета.Очистить();

Стр = ВопросыОтветы.Получить(НомерТекущегоВопроса-1);

З = Новый Запрос;

З.Текст="

|ВЫБРАТЬ

| Ссылка

|ИЗ

| Справочник.Ответы

|ГДЕ Владелец = &Вопрос

|УПОРЯДОЧИТЬ ПО КОД";

З.УстановитьПараметр("Вопрос",Стр.Вопрос);

Выб = З.Выполнить().Выбрать();

Пока Выб.Следующий() Цикл

ВариантОтвета.Добавить(Выб.Ссылка,Выб.Ссылка.Наименование);

КонецЦикла;

Если НЕ Стр.Ответ.Пустая() Тогда

Зн=ВариантОтвета.НайтиПоЗначению(Стр.Ответ);

Зн.Пометка = Истина;

КонецЕсли;

Элементы.ТекстВопроса.Заголовок=Стр.Вопрос.Наименование+Символы.ПС+Стр.Вопрос.ТекстВопроса;

КонецПроцедуры

&НаКлиенте

Процедура ВариантОтветаПометкаПриИзменении(Элемент)

Стр = Элементы.ВариантОтвета.ТекущиеДанные;

Если Стр.Пометка Тогда

Для Каждого Зн Из ВариантОтвета Цикл

Если Стр.Значение=Зн.Значение Тогда Продолжить

КонецЕсли;

Зн.Пометка = Ложь;

КонецЦикла;

Зн = ВопросыОтветы.Получить(НомерТекущегоВопроса - 1);

Зн.Ответ=Стр.Значение;

Иначе

Зн = ВопросыОтветы.Получить(НомерТекущегоВопроса - 1);

Зн.Ответ= Неопределено;

КонецЕсли;

КонецПроцедуры

&НаКлиенте

Процедура Вперед(Команда)

НомерТекущегоВопроса = НомерТекущегоВопроса+1;

ЗапОтветы();

КонецПроцедуры

&НаКлиенте

Процедура Назад(Команда)

НомерТекущегоВопроса = НомерТекущегоВопроса-1;

ЗапОтветы();

КонецПроцедуры

&НаКлиенте

Процедура ЗавершитьТест(Команда)

Оповещение = Новый ОписаниеОповещения("ПослеЗакрытияВопроса",ЭтотОбъект);

ПоказатьВопрос(Оповещение,"Вы действительно хотите закончить тест?",РежимДиалогаВопрос.ДаНет);

КонецПроцедуры

&НаКлиенте

Процедура ПослеЗакрытияВопроса(Результат, Параметры) Экспорт

Если Результат = КодВозвратаДиалога.Нет Тогда Возврат КонецЕсли;

КолПО = 0;

Если НЕ ПроверкаОтветаНаВсеВопросы(КолПО) Тогда Возврат КонецЕсли;

ТестВПроцессе = Ложь;

Элементы.Тест.Видимость = Ложь;

Элементы.РезультатТекст.Заголовок = "Вы ответили правильно на "+КолПО+" вопросов из "+ВопросыОтветы.Количество();

Элементы.РезультатТекст.Видимость = Истина;

ПолучитьДок(ФИО,КолПО);

КонецПроцедуры

&НаСервере

Функция ПолучитьДок(Фио,КолПО)

ДокОбъект = Документы.ПрошедшиеТест.СоздатьДокумент();

ДокОбъект.Дата = ТекущаяДата();

ДокОбъект.ФИО = Фио;

ДокОбъект.КоличествоБаллов = КолПО;

ДокОбъект.Записать(РежимЗаписиДокумента.Проведение,РежимПроведенияДокумента.Неоперативный);

Сообщить("Запись о прохождении теста сделана");

КонецФункции

&НаСервере

Функция ПроверкаОтветаНаВсеВопросы(КолПО)

фл = Истина;

Для Каждого Стр Из ВопросыОтветы Цикл

Если Стр.Ответ.Пустая() Тогда

фл = Ложь;

Сообщить("Вы не ответили на " + Стр.Вопрос.Наименование);

КонецЕсли;

Если Стр.Ответ.ПравильныйОтвет Тогда КолПО = КолПО + 1; КонецЕсли;

КонецЦикла;

Возврат фл;

КонецФункции

&НаКлиенте

Процедура ПриОткрытии(Отказ)

ТестВПроцессе=Ложь;

КонецПроцедуры

&НаКлиенте

Процедура ПередЗакрытием(Отказ, ЗавершениеРаботы, ТекстПредупреждения, СтандартнаяОбработка)

Отказ=ТестВПроцессе;

КонецПроцедуры

* + 1. Модуль формы MainFrom обработки СменаПользователя:

&AtClient

Var CharsMapEnToRu;

&AtClient

Var CharsMapRuToEn;

&AtClient

Var SearchStringQueryToRemember;

&AtClient

Var CountOfFilesToCheck; // Used in asyncronous calls in AddFilesFromDirectoryFindFilesResult

Procedure OnCreateAtServer(Cancel, StandardProcessing)

LoadSettings();

ApplyLanguage();

FillMetadataClasses();

CreateDynamicAttributesAndItems();

FormUniversalListMetadataClassChoiceList();

Obj = FormAttributeToValue("Object");

PictureFolder = New Picture(Obj.GetTemplate("Folder"));

PictureMountedDir = New Picture(Obj.GetTemplate("MountedDir"));

RestoreRootItemsOfTreeOfFiles();

Items.RunAsSecondsLeftToRestorePassword.Visible = False;

RunAsChangePasswordTime\_MinValue = 15;

RunAsChangePasswordTime\_MaxValue = 60;

RunAsChangePasswordOnChangeAtServer();

Try

Items.RunAsUserName.DropListButton = True;

Items.UniversalListMetadataClass.DropListButton = True;

Items.UniversalListMetadataObject.DropListButton = True;

Except

Items.RunAsUserName.ChoiceListButton = True;

Items.UniversalListMetadataClass.ChoiceListButton = True;

Items.UniversalListMetadataObject.ChoiceListButton = True;

EndTry;

EndProcedure

&AtServer

Procedure LoadSettings()

Try

Settings\_Language\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_Language");

Settings\_ChangeCaptionOfTheMainWindow\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_ChangeCaptionOfTheMainWindow");

Settings\_ShowCurrentUsernameInCaption\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_ShowCurrentUsernameInCaption");

Settings\_ShowDatabaseNameInCaption\_Loaded = CommonSettingsStorage.Load("AdministratorsWorkspace", "Settings\_ShowDatabaseNameInCaption");

Settings\_TextToAddToCaption\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_TextToAddToCaption");

Settings\_UseRecent\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_UseRecent");

Settings\_UseHistory\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_UseHistory");

Settings\_UseFavourite\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_UseFavourite");

Settings\_MetadataClassesPanelHeight\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_MetadataClassesPanelHeight");

Settings\_AskBeforeClose\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_AskBeforeClose");

Settings\_AddOnlyReportsAndDataProcessorsToExternalFiles\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_AddOnlyReportsAndDataProcessorsToExternalFiles");

Settings\_DoNotLoadListOfFilesOnStartup\_Loaded = CommonSettingsStorage.Load("СменаПользователя", "Settings\_DoNotLoadListOfFilesOnStartup");

Except

EndTry;

//Default values

If Settings\_Language\_Loaded = Undefined Then

If Metadata.ScriptVariant = Metadata.ObjectProperties.ScriptVariant.English Then

Settings\_Language = "English";

Else

Settings\_Language = "Русский";

EndIf;

Else

Settings\_Language = Settings\_Language\_Loaded;

EndIf;

If Settings\_ChangeCaptionOfTheMainWindow\_Loaded = Undefined Then

Settings\_ChangeCaptionOfTheMainWindow = True;

Else

Settings\_ChangeCaptionOfTheMainWindow = Settings\_ChangeCaptionOfTheMainWindow\_Loaded;

EndIf;

If Settings\_ShowCurrentUsernameInCaption\_Loaded = Undefined Then

Settings\_ShowCurrentUsernameInCaption = True;

Else

Settings\_ShowCurrentUsernameInCaption = Settings\_ShowCurrentUsernameInCaption\_Loaded;

EndIf;

If Settings\_ShowDatabaseNameInCaption\_Loaded = Undefined Then

Settings\_ShowDatabaseNameInCaption = True;

Else

Settings\_ShowDatabaseNameInCaption = Settings\_ShowDatabaseNameInCaption\_Loaded;

EndIf;

If Settings\_TextToAddToCaption\_Loaded = Undefined Then

Settings\_TextToAddToCaption = "";

Else

Settings\_TextToAddToCaption = Settings\_TextToAddToCaption\_Loaded;

EndIf;

If Settings\_UseRecent\_Loaded = Undefined Then

Settings\_UseRecent = False;

Else

Settings\_UseRecent = Settings\_UseRecent\_Loaded;

EndIf;

If Settings\_UseHistory\_Loaded = Undefined Then

Settings\_UseHistory = True;

Else

Settings\_UseHistory = Settings\_UseHistory\_Loaded;

EndIf;

If Settings\_UseFavourite\_Loaded = Undefined Then

Settings\_UseFavourite = True;

Else

Settings\_UseFavourite = Settings\_UseFavourite\_Loaded;

EndIf;

If Settings\_MetadataClassesPanelHeight\_Loaded = Undefined Then

Settings\_MetadataClassesPanelHeight = Items.MetadataClasses.HeightInTableRows;

Else

Settings\_MetadataClassesPanelHeight = Settings\_MetadataClassesPanelHeight\_Loaded;

Items.MetadataClasses.HeightInTableRows = Settings\_MetadataClassesPanelHeight;

EndIf;

If Settings\_AskBeforeClose\_Loaded = Undefined Then

Settings\_AskBeforeClose = False;

Else

Settings\_AskBeforeClose = Settings\_AskBeforeClose\_Loaded;

EndIf;

If Settings\_AddOnlyReportsAndDataProcessorsToExternalFiles\_Loaded = Undefined Then

Settings\_AddOnlyReportsAndDataProcessorsToExternalFiles = False;

Else

Settings\_AddOnlyReportsAndDataProcessorsToExternalFiles = Settings\_AddOnlyReportsAndDataProcessorsToExternalFiles\_Loaded;

EndIf;

If Settings\_DoNotLoadListOfFilesOnStartup\_Loaded = Undefined Then

Settings\_DoNotLoadListOfFilesOnStartup = False;

Else

Settings\_DoNotLoadListOfFilesOnStartup = Settings\_DoNotLoadListOfFilesOnStartup\_Loaded;

EndIf;

//Applying settings

ApplyVisabilityFlags();

EndProcedure

&AtClient

Procedure Settings\_MetadataClassesPanelHeightOnChange(Item)

Items.MetadataClasses.HeightInTableRows = Settings\_MetadataClassesPanelHeight;

SettingsItemChanged(Item);

EndProcedure

&AtServer

Procedure ApplyVisabilityFlags()

If Settings\_UseRecent <> Items.PagesRecent.Visible Then

Items.PagesRecent.Visible = Settings\_UseRecent;

Items.MetadataClasses.VerticalStretch = Not Settings\_UseRecent;

Items.Settings\_MetadataClassesPanelHeight.Enabled = Settings\_UseRecent;

EndIf;

If Settings\_UseHistory <> Items.History.Visible Then

Items.History.Visible = Settings\_UseHistory;

EndIf;

If Settings\_UseFavourite <> Items.Favourite.Visible Then

Items.Favourite.Visible = Settings\_UseFavourite;

EndIf;

EndProcedure

&AtServer

Procedure SaveSettings()

Try

CommonSettingsStorage.Save("СменаПользователя", "Settings\_Language",Settings\_Language);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_ChangeCaptionOfTheMainWindow", Settings\_ChangeCaptionOfTheMainWindow);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_ShowCurrentUsernameInCaption", Settings\_ShowCurrentUsernameInCaption);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_ShowDatabaseNameInCaption", Settings\_ShowDatabaseNameInCaption);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_TextToAddToCaption", Settings\_TextToAddToCaption);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_UseRecent", Settings\_UseRecent);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_UseHistory", Settings\_UseHistory);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_UseFavourite", Settings\_UseFavourite);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_MetadataClassesPanelHeight", Settings\_MetadataClassesPanelHeight);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_AskBeforeClose", Settings\_AskBeforeClose);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_AddOnlyReportsAndDataProcessorsToExternalFiles", Settings\_AddOnlyReportsAndDataProcessorsToExternalFiles);

CommonSettingsStorage.Save("СменаПользователя", "Settings\_DoNotLoadListOfFilesOnStartup", Settings\_DoNotLoadListOfFilesOnStartup);

Except

EndTry;

EndProcedure

&AtClient

Procedure OnOpen(Cancel)

SearchStringQueryToRemember = "";

If NOT Settings\_DoNotLoadListOfFilesOnStartup Then

LoadFileTreeForRootItems();

EndIf;

Settings\_ChangeCaptionOfTheMainWindowOnChange(Undefined);

InitializeTranslateMaps();

#If WebClient Then

Items.RunAsUser.Visible = False;

#EndIf

RunAsChangePassword = True ;

EndProcedure

&AtClient

Procedure LoadFileTreeForRootItems()

RootItems = TreeOfFiles.GetItems();

For Each Item In RootItems Do

Dir = New File(Item.FullFileName);

Dir.BeginCheckingExistence(New NotifyDescription("LoadFileTreeForRootItemsEndChekingDirExistance", ThisForm, Item));

EndDo;

EndProcedure

&AtClient

Procedure LoadFileTreeForRootItemsEndChekingDirExistance(Exist, Item) Export

If Exist Then

AddFilesFromDirectory(Item, Item.FullFileName);

Items.TreeOfFiles.Expand(Item.GetID());

EndIf;

EndProcedure

&AtServer

Function RestoreRootItemsOfTreeOfFiles();

MountedDirectories = Undefined;

Try

MountedDirectories = CommonSettingsStorage.Load("СменаПользователя", "MountedDirectories");

Except

EndTry;

If MountedDirectories <> Undefined Then

RootItems = TreeOfFiles.GetItems();

For Each Struct In MountedDirectories Do

NewRootItem = RootItems.Add();

FillPropertyValues(NewRootItem, Struct);

NewRootItem.Picture = PictureMountedDir;

EndDo;

EndIF;

EndFunction

Procedure FormUniversalListMetadataClassChoiceList()

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "Справочник" ,"Catalog"), ?(UseRu, "Справочники" ,"Catalogs") , , PictureLib.Catalog);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "Документ" ,"Document"), ?(UseRu, "Документы" ,"Documents"), , PictureLib.Document);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "ЖурналДокументов" ,"DocumentJournal"), ?(UseRu, "ЖурналыДокументов" ,"DocumentJournals") , , PictureLib.DocumentJournal);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "ПланВидовХарактеристик" ,"ChartOfCharacteristicTypes"), ?(UseRu, "ПланыВидовХарактеристик" ,"ChartsOfCharacteristicTypes") , , PictureLib.ChartOfCharacteristicTypes);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "ПланСчетов" ,"ChartOfAccounts"), ?(UseRu, "ПланыСчетов" ,"ChartsOfAccounts" ), , PictureLib.ChartOfAccounts);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "ПланВидовРасчета" ,"ChartOfCalculationTypes"), ?(UseRu, "ПланыВидовРасчета" ,"ChartsOfCalculationTypes") , , PictureLib.ChartOfCalculationTypes);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "РегистрСведений" ,"InformationRegister"), ?(UseRu, "РегистрыСведений" ,"InformationRegisters") , , PictureLib.InformationRegister);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "РегистрНакопления" ,"AccumulationRegister"), ?(UseRu, "РегистрыНакопления" ,"AccumulationRegisters"), , PictureLib.AccumulationRegister);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "РегистрБухгалтерии" ,"AccountingRegister"), ?(UseRu, "РегистрыБухгалтерии" ,"AccountingRegisters") , , PictureLib.AccountingRegister);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "РегистрРасчета" ,"CalculationRegister"), ?(UseRu, "РегистрыРасчета" ,"CalculationRegisters") , , PictureLib.CalculationRegister);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "БизнесПроцесс" ,"BusinessProcess"), ?(UseRu, "БизнесПроцессы" ,"BusinessProcesses") , , PictureLib.BusinessProcess);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "Задача" ,"Task"), ?(UseRu, "Задачи" ,"Tasks") , , PictureLib.Task);

Items.UniversalListMetadataClass.ChoiceList.Add(?(UseRu, "ПланОбмена" ,"ExchangePlan"), ?(UseRu, "ПланыОбмена" ,"ExchangePlans") , , PictureLib.ExchangePlan);

EndProcedure

Procedure FormUniversalList(WithPageActivation = False)

ClassName = UniversalListMetadataClass;

ObjectName = UniversalListMetadataObject;

UniversalList.CustomQuery = True;

UniversalList.DynamicDataRead = True;

UniversalList.MainTable = ClassName + "."+ObjectName;

MD = Metadata.FindByFullName(UniversalList.MainTable);

ArrayOfTablesInformation = GetTablesInformation(MD);

HasTables = ArrayOfTablesInformation.Count() > 0;

ArrayOfFieldInformation = New Array;

AddToArrayOfFieldInformation(ArrayOfFieldInformation, "StandardAttributes", MD, HasTables);

AddToArrayOfFieldInformation(ArrayOfFieldInformation, "Attributes", MD, HasTables);

AddToArrayOfFieldInformation(ArrayOfFieldInformation, "Dimensions", MD, HasTables);

AddToArrayOfFieldInformation(ArrayOfFieldInformation, "Resources", MD, HasTables);

AddToArrayOfFieldInformation(ArrayOfFieldInformation, "AddressingAttributes", MD, HasTables);

QueryText = "SELECT "+?(HasTables, "DISTINCT","") + Chars.LF;

For Each FieldInfo In ArrayOfFieldInformation Do

QueryText = QueryText + " MainTable."+FieldInfo.FieldName+","+ Chars.LF;

EndDo;

QueryText = Left(QueryText, StrLen(QueryText)-2);

QueryText = QueryText + Chars.LF +

"FROM

| "+UniversalList.MainTable + " AS MainTable"+ Chars.LF;

For Each TableInfo In ArrayOfTablesInformation Do

QueryText = QueryText + " {LEFT JOIN "+UniversalList.MainTable+"."+ TableInfo.TableName + " AS MainTable"+TableInfo.TableName + " ON (MainTable"+TableInfo.TableName+".Ref = MainTable.Ref)}" +Chars.LF;

EndDo;

If ArrayOfTablesInformation.Count() > 0 Then

QueryText = QueryText +

"

|{WHERE

|";

//MapOfFieldsTitleToChange = New Map;

For Each TableInfo In ArrayOfTablesInformation Do

For Each FieldInfo In TableInfo.ArrayOfFieldInformation Do

FieldAlias = TableInfo.TableName + "\_"+FieldInfo.FieldName;

QueryText = QueryText + " MainTable"+ TableInfo.TableName + "."+FieldInfo.FieldName + ?(FieldInfo.IsRef, ".\*", "") + " AS "+ FieldAlias +"," +Chars.LF;

EndDo;

EndDo;

QueryText = Left(QueryText, StrLen(QueryText)-2);

QueryText = QueryText + "}";

EndIf;

UniversalList.QueryText = QueryText;

Items.DummyUniversalListDefaultPicture.Visible = True;

UniversalListSelectedFields.Clear();

While Items.UniversalList.ChildItems.Count() > 1 Do

Items.Delete(Items.UniversalList.ChildItems[1]);

EndDo;

AlreadyAddedFields = New Array;

ListOfStoredSettings = New ValueList;

RestoredValue = Undefined;

Try

RestoredValue = CommonSettingsStorage.Load("СменаПользователя", "UniversalList-"+ClassName +"-" + ObjectName +"-FieldsSettings");

Except

EndTry;

If RestoredValue <> Undefined Then

ListOfStoredSettings = RestoredValue;

EndIf;

HasVisible = False;

For Each ListValue In ListOfStoredSettings Do

For Each FieldInfo In ArrayOfFieldInformation Do

If FieldInfo.FieldName = ListValue.Value Then

FieldItem = Items.Add("UniversalList" + FieldInfo.FieldName, Type("FormField"), Items.UniversalList);

FieldItem.Type = FormFieldType.InputField;

FieldItem.MultiLine = False;

FieldItem.Height = 1;

FieldItem.DataPath = "UniversalList"+"."+FieldInfo.FieldName;

//FieldItem.FixingInTable = False;

ItemVisible = ListValue.Check;

HasVisible = HasVisible Or ItemVisible;

FieldItem.Visible = ItemVisible;

UniversalListSelectedFields.Add(FieldInfo.FieldName,FieldInfo.FieldName, ItemVisible);

/////

AlreadyAddedFields.Add(FieldInfo.FieldName);

Break;

/////

EndIf;

EndDo;

EndDo;

For Each FieldInfo In ArrayOfFieldInformation Do

If AlreadyAddedFields.Find(FieldInfo.FieldName) <> Undefined Then

Continue;

EndIf;

FieldItem = Items.Add("UniversalList" + FieldInfo.FieldName, Type("FormField"), Items.UniversalList);

FieldItem.Type = FormFieldType.InputField;

FieldItem.MultiLine = False;

FieldItem.Height = 1;

FieldItem.DataPath = "UniversalList"+"."+FieldInfo.FieldName;

//FieldItem.FixingInTable = False;

ItemVisible = True;

HasVisible = True;

FieldItem.Visible = ItemVisible;

UniversalListSelectedFields.Add(FieldInfo.FieldName,FieldInfo.FieldName, ItemVisible);

EndDo;

Items.UniversalList.ChangeRowOrder = True;

Items.DummyUniversalListDefaultPicture.Visible = Not HasVisible;

If WithPageActivation Then

Items.FormPages.CurrentPage = Items.UniversalObjectList;

Items.UniversalObjectListPages.CurrentPage = Items.UniversalListPage;

EndIf;

EndProcedure

Procedure AddToArrayOfFieldInformation(ArrayOfFieldInformation, FieldsType, MD, ExcludeOpenEndedFields = False)

Try

Fields = MD[FieldsType];

Except

Return;

EndTry;

DoNotAddPeriod = False;

ItIsAccountingRegister = False;

If FieldsType = "StandardAttributes" Then

FullMDName = MD.FullName();

Class = Left(FullMDName, Find(FullMDName,".")-1);

DoNotAddPeriod = Class = "РегистрРасчета" or Class = "CalculationRegister";

ItIsAccountingRegister = Class = "РегистрБухгалтерии" or Class = "AccountingRegister";

ElsIf FieldsType = "Dimensions" Or FieldsType = "Resources" Then

FullMDName = MD.FullName();

Class = Left(FullMDName, Find(FullMDName,".")-1);

ItIsAccountingRegister = Class = "РегистрБухгалтерии" or Class = "AccountingRegister";

EndIf;

For Each Field In Fields Do

If DoNotAddPeriod Then

If Field.Name = "Период" OR Field.Name = "Period" Then

Continue;

EndIf;

EndIf;

If ItIsAccountingRegister Then

If FieldsType = "StandardAttributes" Then

If Find(Field.Name, "Субконто") <> 0 OR Find(Field.Name, "ExtDimension") <> 0 Then

Continue;

EndIf;

ElsIf FieldsType = "Dimensions" OR FieldsType = "Resources" Then

If Not Field.Balance Then

Continue;

EndIf;

EndIf;

EndIf;

If Field.Type.ContainsType(Type("ValueStorage")) Then

Continue;

EndIf;

If ExcludeOpenEndedFields Then

If Field.Type.ContainsType(Type("TypeDescription")) Or ( Field.Type.StringQualifiers.AllowedLength = AllowedLength.Variable And Field.Type.StringQualifiers.Length = 0 And Field.Type.ContainsType(Type("String")) ) Then

Continue;

EndIf;

EndIf;

FieldTypes = Field.Type.Types();

IsRef = False;

For Each Type In FieldTypes Do

TypeMD = Metadata.FindByType(Type);

If TypeMD <> Undefined Then

IsRef = True;

Break;

EndIf;

EndDo;

ArrayOfFieldInformation.Add(New Structure("FieldName, IsRef", Field.Name, IsRef));

EndDo;

EndProcedure

Function GetTablesInformation(MD)

ArrayOfTables = New Array;

Tables = Undefined;

Try

Tables = MD.TabularSections;

Except

Return ArrayOfTables;

EndTry;

For Each Table In Tables Do

ArrayOfFieldInformation = New Array;

//AddToArrayOfFieldInformation(ArrayOfFieldInformation, "StandardAttributes", Table);

AddToArrayOfFieldInformation(ArrayOfFieldInformation, "Attributes", Table);

ArrayOfTables.Add(New Structure("TableName, ArrayOfFieldInformation", Table.Name, ArrayOfFieldInformation));

EndDo;

Return ArrayOfTables;

EndFunction

Procedure FillMetadataClasses()

AddToMetadataClasses("Справочники", "Справочник", "Catalogs", "Catalog", PictureLib.Catalog);

AddToMetadataClasses("Документы", "Документ", "Documents", "Document", PictureLib.Document);

AddToMetadataClasses("ЖурналыДокументов", "ЖурналДокументов", "DocumentJournals", "DocumentJournal", PictureLib.DocumentJournal);

AddToMetadataClasses("Отчеты", "Отчет", "Reports", "Report", PictureLib.Report);

AddToMetadataClasses("Обработки", "Обработка", "DataProcessors", "DataProcessor", PictureLib.DataProcessor);

AddToMetadataClasses("ПланыВидовХарактеристик", "ПланВидовХарактеристик", "ChartsOfCharacteristicTypes", "ChartOfCharacteristicTypes", PictureLib.ChartOfCharacteristicTypes);

AddToMetadataClasses("ПланыСчетов", "ПланСчетов", "ChartsOfAccounts", "ChartOfAccounts", PictureLib.ChartOfAccounts);

AddToMetadataClasses("ПланыВидовРасчета", "ПланВидовРасчета", "ChartsOfCalculationTypes", "ChartOfCalculationTypes", PictureLib.ChartOfCalculationTypes);

AddToMetadataClasses("РегистрыСведений", "РегистрСведений", "InformationRegisters", "InformationRegister", PictureLib.InformationRegister);

AddToMetadataClasses("РегистрыНакопления", "РегистрНакопления", "AccumulationRegisters", "AccumulationRegister", PictureLib.AccumulationRegister);

AddToMetadataClasses("РегистрыБухгалтерии", "РегистрБухгалтерии", "AccountingRegisters", "AccountingRegister", PictureLib.AccountingRegister);

AddToMetadataClasses("РегистрыРасчета", "РегистрРасчета", "CalculationRegisters", "CalculationRegister", PictureLib.CalculationRegister);

AddToMetadataClasses("БизнесПроцессы", "БизнесПроцесс", "BusinessProcesses", "BusinessProcess", PictureLib.BusinessProcess);

AddToMetadataClasses("Задачи", "Задача", "Tasks", "Task", PictureLib.Task);

AddToMetadataClasses("ПланыОбмена", "ПланОбмена", "ExchangePlans", "ExchangePlan", PictureLib.ExchangePlan);

AddToMetadataClasses("Константы", "Константа", "Constants", "Constant", PictureLib.Constant);

EndProcedure

Procedure AddToMetadataClasses(ClassNameRu, ClassNameSingularRu, ClassNameEn, ClassNameSingularEn, PictureLink)

NewRow = MetadataClasses.Add();

NewRow.Picture = PictureLink;

NewRow.NameEn = ClassNameEn;

NewRow.NameRu = ClassNameRu;

NewRow.NameSingularEn = ClassNameSingularEn;

NewRow.NameSingularRu = ClassNameSingularRu;

EndProcedure

Procedure CreateDynamicAttributesAndItems()

//Creating new attributes for each metadata class

ArrayOfTypesForTable = new Array;

ArrayOfTypesForTable.Add(Type("ValueTable"));

ArrayOfTypesForPicture = new Array;

ArrayOfTypesForPicture.Add(Type("Picture"));

ArrayOfTypesForString = new Array;

ArrayOfTypesForString.Add(Type("String"));

AttributesArray = new Array;

For Each Row In MetadataClasses Do

ClassName = Row.NameEn;

ClassNameRecent = ClassName + "Recent";

AttributesArray.Add(New FormAttribute(ClassName, New TypeDescription(ArrayOfTypesForTable), "", "", False));

AttributesArray.Add(New FormAttribute("Picture", New TypeDescription(ArrayOfTypesForPicture), ClassName, "", False));

AttributesArray.Add(New FormAttribute("Name", New TypeDescription(ArrayOfTypesForString), ClassName, ?(UseRu, "Имя", "Name"), False));

AttributesArray.Add(New FormAttribute("Synonym", New TypeDescription(ArrayOfTypesForString), ClassName, ?(UseRu, "Синоним", "Synonym"), False));

AttributesArray.Add(New FormAttribute(ClassNameRecent, New TypeDescription(ArrayOfTypesForTable), "", "", False));

AttributesArray.Add(New FormAttribute("Picture", New TypeDescription(ArrayOfTypesForPicture), ClassNameRecent, "", False));

AttributesArray.Add(New FormAttribute("Name", New TypeDescription(ArrayOfTypesForString), ClassNameRecent, ?(UseRu, "Имя", "Name"), False));

AttributesArray.Add(New FormAttribute("Synonym", New TypeDescription(ArrayOfTypesForString), ClassNameRecent, ?(UseRu, "Синоним", "Synonym"), False));

EndDo;

ChangeAttributes(AttributesArray);

//Filling new collections with names and synonyms

For Each Row In MetadataClasses Do

Attribute = ThisForm[Row.NameEn];

For Each ObjectMD in Metadata[Row.NameEn] Do

NewRow = Attribute.Add();

NewRow.Picture = Row.Picture;

NewRow.Name = ObjectMD.Name;

NewRow.Synonym = ObjectMD.Synonym;

EndDo;

Attribute.Sort("Name");

RestoredValue = Undefined;

Try

RestoredValue = CommonSettingsStorage.Load("СменаПользователя", Row.NameEn+"Recent");

Except

EndTry;

If RestoredValue <> Undefined Then

ThisForm[Row.NameEn+"Recent"].Load(RestoredValue);

EndIf;

EndDo;

//Creating new form items (pages and tables)

For Each Row In MetadataClasses Do

ClassName = Row.NameEn;

ClassNameRecent = ClassName + "Recent";

//table of objects

Page = Items.Add("Page"+ClassName, Type("FormGroup"), Items.PagesObjects);

Page.HorizontalStretch = True;

FormTable = Items.Add(ClassName, Type("FormTable"), Page);

FormTable.DataPath = ClassName;

FormTable.Representation = TableRepresentation.List;

FormTable.ReadOnly = True;

FormTable.CommandBarLocation = FormItemCommandBarLabelLocation.None;

FormTable.SetAction("Selection", "ObjectTableSelection");

FormTable.EnableStartDrag = True;

FormTable.EnableDrag = False;

FormTable.HorizontalStretch = True;

ContextMenuCommand = Items.Add(ClassName+"ContextMenuOpenInUniversalList", Type("FormButton"), FormTable.ContextMenu);

ContextMenuCommand.CommandName = "OpenInUniversalObjectList";

If UseRu Then

ContextMenuCommand.Title = "Открыть в универсальном списке";

EndIf;

ContextMenuCommand = Items.Add(ClassName+"ContextMenuOpenSelectedForm", Type("FormButton"), FormTable.ContextMenu);

ContextMenuCommand.CommandName = "OpenSelectedForm";

If UseRu Then

ContextMenuCommand.Title = "Открыть произвольную форму";

EndIf;

ColumnGroup = Items.Add(ClassName + "Group", Type("FormGroup"), FormTable);

ColumnGroup.Group = ColumnsGroup.InCell;

PictureItem = Items.Add(ClassName + "Picture", Type("FormField"), ColumnGroup);

PictureItem.Type = FormFieldType.PictureField;

PictureItem.DataPath = ClassName+".Picture";

PictureItem.ShowInHeader = False;

NameItem = Items.Add(ClassName + "Name", Type("FormField"), ColumnGroup);

NameItem.Type = FormFieldType.InputField;

NameItem.DataPath = ClassName+".Name";

SynonymItem = Items.Add(ClassName + "Synonym", Type("FormField"), FormTable);

SynonymItem.Type = FormFieldType.InputField;

SynonymItem.DataPath = ClassName+".Synonym";

//table of recent objects

Page = Items.Add("Page"+ClassNameRecent, Type("FormGroup"), Items.PagesRecent);

Page.VerticalStretch = True;

FormTable = Items.Add(ClassNameRecent, Type("FormTable"), Page);

FormTable.DataPath = ClassNameRecent;

FormTable.Representation = TableRepresentation.List;

FormTable.ReadOnly = True;

FormTable.CommandBarLocation = FormItemCommandBarLabelLocation.None;

FormTable.Header = True;

FormTable.TitleLocation = FormItemTitleLocation.None;

FormTable.Title = ?(UseRu, "Недавние", "Recent");

FormTable.SetAction("Selection", "ObjectTableSelection");

FormTable.EnableStartDrag = True;

FormTable.EnableDrag = False;

FormTable.VerticalStretch = True;

ContextMenuCommand = Items.Add(ClassName+"Recent"+"ContextMenuOpenInUniversalList", Type("FormButton"), FormTable.ContextMenu);

ContextMenuCommand.CommandName = "OpenInUniversalObjectList";

If UseRu Then

ContextMenuCommand.Title = "Открыть в универсальном списке";

EndIf;

ContextMenuCommand = Items.Add(ClassName+"Recent"+"ContextMenuOpenSelectedForm", Type("FormButton"), FormTable.ContextMenu);

ContextMenuCommand.CommandName = "OpenSelectedForm";

If UseRu Then

ContextMenuCommand.Title = "Открыть произвольную форму";

EndIf;

ColumnGroup = Items.Add(ClassNameRecent + "Group", Type("FormGroup"), FormTable);

ColumnGroup.Group = ColumnsGroup.InCell;

PictureItem = Items.Add(ClassNameRecent + "Picture", Type("FormField"), ColumnGroup);

PictureItem.Type = FormFieldType.PictureField;

PictureItem.DataPath = ClassNameRecent+".Picture";

PictureItem.ShowInHeader = False;

NameItem = Items.Add(ClassNameRecent + "Name", Type("FormField"), ColumnGroup);

NameItem.Type = FormFieldType.InputField;

NameItem.DataPath = ClassNameRecent+".Name";

NameItem.Title = ?(UseRu, "Недавние", "Recent");

//SynonymItem = Items.Add(ClassNameRecent + "Synonym", Type("FormField"), FormTable);

//SynonymItem.Type = FormFieldType.InputField;

//SynonymItem.DataPath = ClassNameRecent+".Synonym";

EndDo;

EndProcedure

Procedure ApplyLanguage()

UseRu = (Settings\_Language = "Русский");

//LanguageCode = ?(UseRu, "ru", "en");

Items.MetadataClassesNameEn.Visible = NOT UseRu;

Items.MetadataClassesNameRu.Visible = UseRu;

Items.Settings\_Language.Title = "Language / Язык";

Items.MetadataClassesNameEn.Title = "Metadata class";

Items.MetadataClassesNameRu.Title = "Класс метаданных";

Items.SearchString.Title = ?(UseRu, "Поиск", "Search");

Items.DatabaseObjects.Title = ?(UseRu, "Объекты базы данных", "Database objects");

Items.ExternalFiles.Title = ?(UseRu, "Внешние файлы", "External files");

Items.RunAsUser.Title = ?(UseRu, "Запуск от имени пользователя", "Run as user");

Items.UniversalObjectList.Title = ?(UseRu, "Универсальный динамический список", "Universal dynamic list");

Items.SettingsAndAbout.Title = ?(UseRu, "Настройки", "Settings");

Items.Close.Title = ?(UseRu, "Закрыть", "Close");

//DatabaseObjects page

Items.HistoryPresentation.Title = ?(UseRu, "История", "History");

Items.FavouriteName.Title = ?(UseRu, "Избранное (перетащите для добавления)", "Favourite (drag & drop to add an object)");

Items.FavouriteContextMenuOpenInUniversalObjectList.Title = ?(UseRu, "Открыть в универсальном списке", "Open in universal dynamic list");

Items.FavouriteContextMenuOpenSelectedForm.Title = ?(UseRu, "Открыть произвольную форму", "Open selected form");

//Universal dynamic list

Items.UniversalListMetadataClass.Title = ?(UseRu, "Класс" , "Class");

Items.UniversalListMetadataObject.Title = ?(UseRu, "Объект" , "Object");

Items.UniversalListPage.Title = ?(UseRu, "Динамический список" , "Dynamic list");

Items.UniversalListFieldsPage.Title = ?(UseRu, "Выбранные поля" , "Selected fields");

Items.UniversalListFilterPage.Title = ?(UseRu, "Отбор" , "Filter");

Items.UniversalListOpenInSelectedFormCM.Title = ?(UseRu, "Открыть ссылку в выбранной форме" , "Open reference in selected form");

Items.UniversalListAddToFavourite.Title = ?(UseRu, "Добавить ссылку в избранное" , "Add reference to favourite");

Items.UniversalListAddToFavouriteCM.Title = ?(UseRu, "Добавить ссылку в избранное" , "Add reference to favourite");

//External files

Items.TreeOfFilesAdd.Title = ?(UseRu, "Добавить каталог" , "Add catalog");

Items.TreeOfFilesChange.Title = ?(UseRu, "Изменить название" , "Change caption");

Items.TreeOfFilesDelete.Title = ?(UseRu, "Убрать из списка" , "Remove from list");

Items.TreeOfFilesHierarchicalList.Title = ?(UseRu, "Показывать списком" , "List view");

Items.TreeOfFilesTree.Title = ?(UseRu, "Показывать деревом" , "Tree view");

Items.TreeOfFilesExternalFilesUpdate.Title = ?(UseRu, "Обновить список" , "Update list");

Items.TreeOfFilesExternalFilesAddToFavourite.Title = ?(UseRu, "Добавить в избранное" , "Add to favourite");

Items.TreeOfFilesAddCM.Title = ?(UseRu, "Добавить каталог" , "Add catalog");

Items.TreeOfFilesChangeCM.Title = ?(UseRu, "Изменить название" , "Change caption");

Items.TreeOfFilesDeleteCM.Title = ?(UseRu, "Убрать из списка" , "Remove from list");

Items.TreeOfFilesHierarchicalListCM.Title = ?(UseRu, "Показывать списком" , "List view");

Items.TreeOfFilesTreeCM.Title = ?(UseRu, "Показывать деревом" , "Tree view");

Items.TreeOfFilesExternalFilesUpdateCM.Title = ?(UseRu, "Обновить список" , "Update list");

Items.TreeOfFilesExternalFilesAddToFavouriteCM.Title = ?(UseRu, "Добавить в избранное" , "Add to favourite");

//RunAs page

Items.RunAsUserName.Title = ?(UseRu, "Имя пользователя" , "Username");

Items.RunAsCurrentPasswordHash.Title = ?(UseRu, "Хэш пароля" , "Password hash");

Items.RunAsChangePassword.Title = ?(UseRu, "Изменять пароль пользователя при запуске (временно)" , "Change user's password on start (temporarily)");

Items.ChangePasswordDecoration.Title = ?(UseRu, "менять пароль на ""123"" на" , "сhange password to ""123"" for");

Items.RunAsChangePasswordTime.Title = ?(UseRu, "секунд" , "seconds");

Items.RunAsRun.Title = ?(UseRu, "Запустить новый сеанс" , "Run application");

Items.RunAsSecondsLeftToRestorePassword.Title = ?(UseRu, "Осталось секунд" , "seconds left");

Items.RunAsSecondsLeftToRestorePassword.TitleLocation = FormItemTitleLocation.Left;

Items.RunAsAdditionalParams.Title = ?(UseRu, "Дополнительные параметры запуска (параметры командной строки)", "Additional startup option (command line parameters)");

//Settings

Items.Settings\_ChangeCaptionOfTheMainWindow.Title = ?(UseRu, "Изменять заголовок главного окна при запуске" , "Сhange caption of the main window on start");

Items.Settings\_ShowCurrentUsernameInCaption.Title = ?(UseRu, "Показывать имя пользователя" , "Show current username");

Items.Settings\_ShowDatabaseNameInCaption.Title = ?(UseRu, "Показывать имя базы данных" , "Show database name");

Items.Settings\_TextToAddToCaptionTitle.Title = ?(UseRu, "Добавлять следующий текст :" , "Add this text to caption :");

Items.Settings\_TextToAddToCaption.Title = ?(UseRu, "Добавлять следующий текст к заголовку" , "Add this text to caption :");

Items.Settings\_UseRecent.Title = ?(UseRu, "Показывать недавние" , "Use recent");

Items.Settings\_UseHistory.Title = ?(UseRu, "Показывать историю" , "Use history");

Items.Settings\_UseFavourite.Title = ?(UseRu, "Показывать избранное" , "Use favourite");

Items.Settings\_MetadataClassesPanelHeight.Title = ?(UseRu,"Высота панели классов объектов" , "Metadata classes panel height");

Items.ClearHistory.Title = ?(UseRu, "Очистить историю" , "Clear history");

Items.Settings\_AskBeforeClose.Title = ?(UseRu, "Предотвращать случайное закрытие (спрашивать перед закрытием)" , "Prevent occasional closing (ask before close)");

Items.Settings\_AddOnlyReportsAndDataProcessorsToExternalFiles.Title = ?(UseRu, "Добавлять во внешние файлы только отчеты и обработки (файлы .erf и .epf)" , "Add only reports and data processors to external files (.erf and .epf files)");

Items.Settings\_DoNotLoadListOfFilesOnStartup.Title = ?(UseRu, "Загружать список внешних файлов только при нажатии на кнопку ""Обновить"", не загружать при запуске (делает запуск быстрее)" , "Load list of external files only by pressing ""Update"" button, do not load on start (allows to start faster)");

Items.ProjectLink.Title = ?(UseRu, "Владимир Литвиненко, 2013. Страница обработки: http://infostart.ru/public/239307/" , "Vladimir Litvinenko, 2013. Project page: http://infostart.ru/public/239307/");

EndProcedure

&AtClient

Procedure MetadataClassesOnActivateRow(Item)

CD = Items.MetadataClasses.CurrentData;

If CD <> Undefined Then

Items.PagesObjects.CurrentPage = Items["Page"+CD.NameEn];

Items.PagesRecent.CurrentPage = Items["Page"+CD.NameEn+"Recent"];

EndIf;

EndProcedure

&AtClient

Procedure ObjectTableSelection(Item, SelectedRow, Field, StandardProcessing)

ClassCD = Items.MetadataClasses.CurrentData;

ObjectCD = Item.CurrentData;

If ClassCD = Undefined Or ObjectCD = Undefined Then

Return;

EndIf;

OpenDefaultForm(ClassCD.NameSingularEn, ObjectCD.Name);

If Right(Item.Name, StrLen("Recent")) <> "Recent" Then

AddToRecent(ClassCD.NameEn, ClassCD.NameSingularEn, ObjectCD.Picture, ObjectCD.Name, ObjectCD.Synonym);

EndIf;

AddToHistory(ClassCD.NameSingularEn, ObjectCD.Picture, ObjectCD.Name, "");

EndProcedure

&AtClient

Procedure OpenDefaultForm(ClassName, ObjectName)

If ClassName = "Constant" Then

ConstantEditorFormName = StrReplace(FormName,"MainForm", "ConstantEditor");

OpenForm(ConstantEditorFormName, New Structure("ConstantName, UseRu", ObjectName, UseRu));

ElsIf ClassName = "Report" or ClassName = "DataProcessor" Then

OpenForm(ClassName+"."+ObjectName+".Form");

Else

OpenForm(ClassName+"."+ObjectName+".ListForm");

EndIf;

EndProcedure

&AtClient

Procedure AddToRecent(NameEn, NameSingularEn, Picture, Name, Synonym)

Table = ThisForm[NameEn+"Recent"];

ArrayToDelete = New Array;

For Each row In Table Do

If Row.Picture = Picture And Row.Name = Name Then

ArrayToDelete.Add(Row);

EndIf;

EndDo;

For Each Row In ArrayToDelete Do

Table.Delete(Row);

EndDo;

NewRow = Table.Insert(0);

//NewRow.Class = NameSingularEn;

NewRow.Picture = Picture;

NewRow.Name = Name;

NewRow.Synonym = Synonym;

If Table.Count() > 10 Then

Table.Delete(10);

EndIf;

SaveRecent(NameEn+"Recent");

EndProcedure

&AtServer

Procedure SaveRecent(AttributeName);

Try

CommonSettingsStorage.Save("СменаПользователя", AttributeName, ThisForm[AttributeName].Unload());

Except

EndTry;

EndProcedure

&AtClient

Procedure AddToHistory(NameSingularEn, Picture, Name, FormName, FullFileName = Undefined, Ref = Undefined)

NewRow = History.Insert(0);

NewRow.Picture = Picture;

NewRow.Name = Name;

NewRow.Class = NameSingularEn;

NewRow.FormName = FormName;

If FullFileName <> Undefined Then

NewRow.FullFileName = FullFileName;

NewRow.Presentation = Name+Chars.LF+?(UseRu, "Открыть файл '"+Name+"'", "Open file '"+Name+"'");

ElsIf FormName = "" Then

NewRow.Presentation = Name+Chars.LF+?(UseRu, "Открыть форму по-умолчанию", "Open default form");

ElsIf FormName = "UniversalDynamicList" Then

NewRow.Presentation = Name+Chars.LF+?(UseRu, "Открыть в универсальном списке", "Open in universal dynamic list");

Else

If UseRu Then

NewRow.Presentation = Name+Chars.LF+"Открыть форму '"+FormName+"'";

Else

NewRow.Presentation = Name+Chars.LF+"Open '"+FormName+"' form";

EndIf;

EndIf;

If History.Count() > 50 Then

History.Delete(50);

EndIf;

EndProcedure

&AtClient

Procedure FavouriteDrag(Item, DragParameters, StandardProcessing, Row, Field)

StandardProcessing = False;

ClassCD = Items.MetadataClasses.CurrentData;

If ClassCD = Undefined Then

Return;

EndIf;

Try

For Each Row In DragParameters.Value Do

Picture = Row.Picture;

Name = Row.Name;

NewRow = Favourite.Insert(0);

NewRow.ClassEn = ClassCD.NameSingularEn;

NewRow.ClassRu = ClassCD.NameSingularRu;

NewRow.Name = Name;

NewRow.Picture = Picture;

Items.Favourite.CurrentRow = NewRow.GetID();

EndDo;

Except

EndTry;

EndProcedure

&AtClient

Procedure FavouriteBeforeAddRow(Item, Cancel, Clone, Parent, Folder)

Cancel = True;

EndProcedure

&AtClient

Procedure FavouriteSelection(Item, SelectedRow, Field, StandardProcessing)

CD = Item.CurrentData;

If ValueIsFilled(CD.Ref) Then

TableName = GetFullMetadataNameForLink(CD.Ref);

OpenForm(TableName+".ObjectForm",New Structure("Key, Ключ", CD.Ref, CD.Ref));

ElsIf Not IsBlankString(CD.FullFileName) Then

If TryToOpenFile(CD.Name, CD.FullFileName) Then

AddToHistory("", CD.Picture, CD.Name, "", CD.FullFileName);

EndIf;

Else

OpenDefaultForm(CD.ClassEn, CD.Name);

AddToHistory(CD.ClassEn, CD.Picture, CD.Name, "");

EndIf;

EndProcedure

&AtClient

Procedure HistorySelection(Item, SelectedRow, Field, StandardProcessing)

CD = Item.CurrentData;

If Not IsBlankString(CD.FullFileName) Then

TryToOpenFile(CD.Name, CD.FullFileName);

ElsIf CD.FormName = "" Then

OpenDefaultForm(CD.Class, CD.Name);

ElsIf

CD.FormName = "UniversalDynamicList" Then

UniversalListMetadataClass = CD.Class;

UniversalListMetadataClassOnChange(Undefined);

UniversalListMetadataObject = CD.Name;

FormUniversalList(True);

Else

OpenForm(CD.Class+"."+CD.Name+".Form."+CD.FormName);

EndIf;

EndProcedure

&AtClient

Procedure OpenSelectedForm(Command)

Class = "";

Name = "";

Picture = Undefined;

Item = ThisForm.CurrentItem;

If Item = Items.Favourite Then

CD = Items.Favourite.CurrentData;

If CD = Undefined Then

Return;

EndIf;

If ValueIsFilled(CD.Ref) Then

TableName = GetFullMetadataNameForLink(CD.Ref);

FormList = GetFormListByFullName(TableName);

If FormList = Undefined Then

Message = ?(UseRu,"Для этого объекта не определены формы", "There is no form defined for this object");

ShowUserNotification(Message,,Message,);

Return;

EndIf;

AdditionalParams = New Structure("TableName, CurrentRow", TableName, CD.Ref);

ShowChooseFromList(New NotifyDescription("UniversalListOpenInSelectedFormHandleSelection", ThisForm, AdditionalParams), FormList, ThisForm.CurrentItem);

Return;

Else

Class = ?(UseRu, CD.ClassRu, CD.ClassEn);

Name = CD.Name;

Picture = CD.Picture;

EndIf;

Else

ClassCD = Items.MetadataClasses.CurrentData;

ObjectCD = Item.CurrentData;

If ClassCD = Undefined or ObjectCD = Undefined Then

Return;

EndIf;

Class = ClassCD.NameSingularEn;

Name = ObjectCD.Name;

Picture = ObjectCD.Picture;

EndIf;

FormList = GetFormList(Class, Name);

If FormList = Undefined Then

Message = ?(UseRu, "Для этого объекта не определены формы", "There is no form defined for this object");

ShowUserNotification(Message,,Message,);

Return;

EndIf;

AdditionalParams = New Structure("Class, Name, Picture", Class, Name, Picture);

ShowChooseFromList(New NotifyDescription("OpenSelectedFormFormHandleSelection", ThisForm, AdditionalParams), FormList, ThisForm.CurrentItem);

EndProcedure

&AtClient

Procedure OpenSelectedFormFormHandleSelection(SelectedValue, AdditionalParams) Export

If SelectedValue <> Undefined Then

Try

OpenForm(AdditionalParams.Class+"."+AdditionalParams.Name+".Form."+SelectedValue.Value);

AddToHistory(AdditionalParams.Class, AdditionalParams.Picture, AdditionalParams.Name, SelectedValue.Value);

Except

Message = ?(UseRu,"Эта форма не может быть открыта в текущем режиме запуска 1С:Предприятия","This form cannot be opened in current running mode of 1C:Enterprise");

ShowUserNotification(Message,,Message,);

Return;

EndTry;

EndIf;

EndProcedure

&AtClient

Procedure OpenInUniversalObjectList(Command)

Class = "";

Name = "";

Picture = Undefined;

Item = ThisForm.CurrentItem;

If Item = Items.Favourite Then

CD = Items.Favourite.CurrentData;

If CD = Undefined Then

Return;

EndIf;

Class = ?(UseRu, CD.ClassRu, CD.ClassEn);

Name = CD.Name;

Picture = CD.Picture;

Else

ClassCD = Items.MetadataClasses.CurrentData;

ObjectCD = Item.CurrentData;

If ClassCD = Undefined or ObjectCD = Undefined Then

Return;

EndIf;

Class = ?(UseRu, ClassCD.NameSingularRu, ClassCD.NameSingularEn);

Name = ObjectCD.Name;

Picture = ObjectCD.Picture;

EndIf;

If Items.UniversalListMetadataClass.ChoiceList.FindByValue(Class) = Undefined Then

Message = ?(UseRu,"Универсальный список не может быть открыт для этого объекта", "Universal list cannot be opened for this object");

ShowUserNotification(Message,,Message,);

Return;

EndIf;

UniversalListMetadataClass = Class;

UniversalListMetadataClassOnChange(Undefined);

UniversalListMetadataObject = Name;

MDClassCurrentRow = Items.MetadataClasses.CurrentRow;

FormUniversalList(True);

AddToHistory(Class, Picture, Name, "UniversalDynamicList");

Items.MetadataClasses.CurrentRow = MDClassCurrentRow;

EndProcedure

&AtServerNoContext

Function GetFormList(Class, Name)

Return GetFormListByFullName(Class+"."+Name);

EndFunction

&AtServerNoContext

Function GetFormListByFullName(FullName)

MD = Metadata.FindByFullName(FullName);

If MD = Undefined Then

Return Undefined;

EndIf;

If MD.Forms.Count() = 0 Then

Return Undefined;

EndIf;

List = New ValueList;

For Each Form In MD.Forms Do

List.Add(Form.Name);

EndDo;

Return List;

EndFunction

&AtClient

Procedure UniversalListSelectedFieldsBeforeAddRow(Item, Cancel, Clone, Parent, Folder)

Cancel = True;

EndProcedure

&AtClient

Procedure UniversalListSelectedFieldsBeforeDeleteRow(Item, Cancel)

Cancel = True;

EndProcedure

&AtClient

Procedure UniversalListMetadataClassClearing(Item, StandardProcessing)

StandardProcessing = False;

EndProcedure

&AtClient

Procedure UniversalListMetadataObjectClearing(Item, StandardProcessing)

StandardProcessing = False;

EndProcedure

&AtClient

Procedure UniversalListMetadataClassOnChange(Item)

If UniversalListMetadataClass = UniversalListMetadataClassPrev Then

Return;

EndIf;

Items.UniversalListMetadataObject.ChoiceList.Clear();

UniversalListMetadataObject = "";

If UniversalListMetadataClass <> "" Then

UniversalListMetadataClassOnChangeAtServer();

EndIf;

UniversalListMetadataClassPrev = UniversalListMetadataClass;

EndProcedure

Procedure UniversalListMetadataClassOnChangeAtServer()

MDCollection = Metadata[Items.UniversalListMetadataClass.ChoiceList.FindByValue(UniversalListMetadataClass).Presentation];

For Each MD In MDCollection Do

Items.UniversalListMetadataObject.ChoiceList.Add(MD.Name);

EndDo;

Items.UniversalListMetadataObject.ChoiceList.SortByValue();

EndProcedure

&AtClient

Procedure UniversalListMetadataObjectOnChange(Элемент)

If UniversalListMetadataObject = UniversalListMetadataObjectPrev Then

Return;

EndIf;

If UniversalListMetadataObject <> "" Then

FormUniversalList();

EndIf;

UniversalListMetadataObjectPrev = UniversalListMetadataObject;

EndProcedure

&AtClient

Procedure UniversalListSelectedFieldsOnChange(Item)

UniversalListSaveSelectedFieldsOnExitOrPageChange = True;

//UniversalListSelectedFieldsOnChangeAtServer();

EndProcedure

&AtServer

Procedure UniversalListSelectedFieldsOnChangeAtServer()

CurrentRow = Items.UniversalList.CurrentRow;

Try

CommonSettingsStorage.Save("СменаПользователя", "UniversalList-"+StrReplace(UniversalList.MainTable,".","-")+"-FieldsSettings", UniversalListSelectedFields);

FormUniversalList();

If ValueIsFilled(CurrentRow) Then

Items.UniversalList.CurrentRow = CurrentRow;

EndIf;

Except

EndTry;

EndProcedure

&AtClient

Procedure OnClose(ApplicationIsSuttingDown = Undefined)

If ApplicationIsSuttingDown <> True Then

If UniversalListSaveSelectedFieldsOnExitOrPageChange Then

UniversalListSelectedFieldsOnChangeAtServer();

EndIf;

SaveSettings();

EndIf;

EndProcedure

&AtClient

Procedure UniversalObjectListPagesOnCurrentPageChange(Item, CurrentPage)

If UniversalListSaveSelectedFieldsOnExitOrPageChange Then

UniversalListSelectedFieldsOnChangeAtServer();

UniversalListSaveSelectedFieldsOnExitOrPageChange = False;

EndIf;

EndProcedure

&AtClient

Procedure UniversalListMetadataObjectOpening(Item, StandardProcessing)

StandardProcessing = False;

If UniversalListMetadataObject <> "" And UniversalListMetadataClass <> "" Then

OpenForm(UniversalListMetadataClass+"."+UniversalListMetadataObject+".ListForm");

EndIf;

EndProcedure

&AtClient

Procedure UniversalListOpenInSelectedForm(Command)

CR = Items.UniversalList.CurrentRow;

If Not ValueIsFilled(CR) Then

Message = ?(UseRu,"Cначала выделите строку", "Select row first");

ShowUserNotification(Message,,Message,);

Return;

EndIf;

TableName = GetFullMetadataNameForLink(CR);

FormList = GetFormListByFullName(TableName);

If FormList = Undefined Then

Message = ?(UseRu,"Для этого объекта не определены формы", "There is no form defined for this object");

ShowUserNotification(Message,,Message,);

Return;

EndIf;

AdditionalParams = New Structure("TableName, CurrentRow", TableName, Items.UniversalList.CurrentRow);

ShowChooseFromList(New NotifyDescription("UniversalListOpenInSelectedFormHandleSelection", ThisForm, AdditionalParams), FormList, ThisForm.CurrentItem);

EndProcedure

&AtClient

Procedure UniversalListOpenInSelectedFormHandleSelection(SelectedValue, AdditionalParams) Export

If SelectedValue <> Undefined Then

Try

CR = AdditionalParams.CurrentRow;

OpenForm(AdditionalParams.TableName+".Form."+SelectedValue.Value, New Structure("Key, Ключ, CurrentRow, ТекущаяСтрока", CR, CR, CR, CR));

Except

Message = ?(UseRu,"Эта форма не может быть открыта в текущем режиме запуска 1С:Предприятия","This form cannot be opened in current running mode of 1C:Enterprise");

ShowUserNotification(Message,,Message,);

Return;

EndTry;

EndIf;

EndProcedure

&AtClient

Procedure UniversalListAddToFavourite(Command)

CR = Items.UniversalList.CurrentRow;

If Not ValueIsFilled(CR) Then

Message = ?(UseRu,"Cначала выделите строку", "Select row first");

ShowUserNotification(Message,,Message,);

Return;

EndIf;

If Find(String(TypeOf(CR)), "Регистр") <> 0 Or Find(String(TypeOf(CR)), "Register") <> 0 Then

Message = ?(UseRu,"Этот объект не содержит сслыки и не может быть добавлен в избранное", "This object does not have reference and cannot be added to favorites");

ShowUserNotification(Message,,Message,);

Return;

EndIf;

Picture = PictureLib.Form;

NewRow = Favourite.Insert(0);

NewRow.Name = String(CR);

NewRow.Ref = CR;

NewRow.Picture = Picture;

Items.Favourite.CurrentRow = NewRow.GetID();

EndProcedure

&AtServerNoContext

Function GetFullMetadataNameForLink(Link)

Return Link.Metadata().FullName();

EndFunction

&AtClient

Procedure DummyBeforeAddRow(Item, Cancel, Clone, Parent, Folder)

Cancel = True;

EndProcedure

&AtClient

Procedure DummyBeforeDeleteRow(Item, Cancel)

Cancel = True;

EndProcedure

&AtClient

Procedure TreeOfFilesBeforeAddRow(Item, Cancel, Clone, Parent, Folder)

Cancel = True;

EndProcedure

&AtClient

Procedure AddFilesFromDirectory(Parent, Directory)

BeginFindingFiles(New NotifyDescription("AddFilesFromDirectoryFindFilesResult", ThisForm, Parent), Directory, "\*", False);

EndProcedure

&AtClient

Procedure AddFilesFromDirectoryFindFilesResult(ArrayOfFiles, Parent) Export

Folders = New ValueList;

Files = New ValueList;

If ArrayOfFiles.Count() = 0 Then

Return;

EndIf;

IndexToCheck = 0;

Params = New Structure("Parent, Folders, Files, IndexToCheck, ArrayOfFiles", Parent, Folders, Files, IndexToCheck, ArrayOfFiles);

ArrayOfFiles[Params.IndexToCheck].BeginCheckingIsDirectory(New NotifyDescription("CheckNextFileIsDirectoryResult", ThisForm, Params));

EndProcedure

&AtClient

Procedure CheckNextFileIsDirectoryResult(IsDirectory, Params) Export

ArrayOfFiles = Params.ArrayOfFiles;

CheckedFile = ArrayOfFiles[Params.IndexToCheck];

If IsDirectory Then

Params.Folders.Add(CheckedFile, CheckedFile.Name);

Else

Params.Files.Add(CheckedFile, CheckedFile.Name);

EndIf;

// Switching to next file

Params.IndexToCheck = Params.IndexToCheck + 1;

// If checked all files then just calling final method to add files in tree

If Params.IndexToCheck = ArrayOfFiles.Count() Then

AddFilesFromDirectoryEnd(Params.Parent, Params.Folders, Params.Files);

Else

ArrayOfFiles[Params.IndexToCheck].BeginCheckingIsDirectory(New NotifyDescription("CheckNextFileIsDirectoryResult", ThisForm, Params));

EndIf;

EndProcedure

&AtClient

Procedure AddFilesFromDirectoryEnd(Parent, Folders, Files) Export

Folders.SortByPresentation();

Files.SortByPresentation();

For Each Value In Folders Do

Folder = Value.Value;

NewItem = Parent.GetItems().Add();

NewItem.Alias = Folder.Name;

NewItem.FullFileName = Folder.FullName;

NewItem.Picture = PictureFolder;

AddFilesFromDirectory(NewItem, Folder.FullName);

EndDo;

For Each Value In Files Do

File = Value.Value;

Extension = Upper(File.Extension);

If Settings\_AddOnlyReportsAndDataProcessorsToExternalFiles Then

If Extension <> ".EPF" And Extension <> ".ERF" Then

Continue;

EndIf;

EndIf;

NewItem = Parent.GetItems().Add();

NewItem.Alias = File.Name;

NewItem.FullFileName = File.FullName;

NewItem.IsFile = True;

NewItem.Extension = Extension;

If Extension = ".EPF" Then

NewItem.Picture = PictureLib.DataProcessor;

ElsIf Extension = ".ERF" Then

NewItem.Picture = PictureLib.Report;

Else

NewItem.Picture = PictureLib.Document;

EndIf;

EndDo;

EndProcedure

&AtClient

Procedure ExternalFilesEditAlias(Command)

CD = Items.TreeOfFiles.CurrentData;

IsTopItem = CD.GetParent() = Undefined;

//alias can be changed only for top items

If Not IsTopItem Then

Message = "Alias can be changed only for top items (added catalogs)";

ShowUserNotification(Message, , Message,);

Return;

EndIf;

CD.CatalogPath = " ("+CD.FullFileName+")";

EndProcedure

&AtClient

Procedure TreeOfFilesSelection(Item, SelectedRow, Field, StandardProcessing)

CD = Items.TreeOfFiles.CurrentData;

If CD = Undefined Then

Return;

EndIf;

If NOT CD.IsFile Then

Return;

EndIf;

Alias = CD.Alias;

Extension = CD.Extension;

FullFileName = CD.FullFileName;

If TryToOpenFile(Alias, FullFileName, Extension) Then

AddToHistory("", CD.Picture, CD.Alias, "", CD.FullFileName);

EndIf;

EndProcedure

&AtClient

Function TryToOpenFile(Alias, FullFileName, Extension = Undefined)

File = New File(FullFileName);

If Extension = Undefined Then

Length = StrLen(FullFileName);

K = Length;

PeriodFounded = False;

While K > 1 Do

If Mid(FullFileName, K, 1) = "." Then

PeriodFounded = True;

Break;

EndIf;

K = K -1;

EndDo;

If PeriodFounded Then

Extension = Upper(Right(FullFileName, Length - K + 1));

EndIf;

EndIf;

If Extension = ".EPF" Or Extension = ".ERF" Then

IsReport = Extension = ".ERF";

Try

#If ThickClientOrdinaryApplication Then

If IsReport Then

FormToOpen = ExternalReports.GetForm(FullFileName);

Else

FormToOpen = ExternalDataProcessors.GetForm(FullFileName);

EndIf;

If FormToOpen <> Undefined Then

FormToOpen.Open();

Else

If IsReport Then

ExternalReport = ExternalReports.Create(FullFileName, False);

FormToOpen = ExternalReport.GetForm();

If FormToOpen <> Undefined Then

FormToOpen.Open();

EndIf;

Else

ExternalDP = ExternalDataProcessors.Create(FullFileName, False);

FormToOpen = ExternalDP.GetForm();

If FormToOpen <> Undefined Then

FormToOpen.Open();

EndIf;

EndIf;

EndIf;

#Else

//managed application

StorageAddress = "";

StorageAddress = PutToTempStorage(New BinaryData(FullFileName), ThisForm.UUID);

ConnectedName = ConnectExternalProcessorReport(StorageAddress, IsReport);

OpenForm(?(IsReport, "ExternalReport.","ExternalDataProcessor.")+ ConnectedName +".Form");

#EndIf

Except

Message = ?(UseRu,

"Выбранный файл не может быть открыт в этой конфигурации или предназначен для другого режима запуска 1С:Предприятия",

"Selected file cannot be opened in this configuration or is intended for another run mode of 1C:Enterprise");

ShowUserNotification(Message, , Message);

Return False;

EndTry;

ElsIf Extension = ".MXL" Then

Document = New SpreadsheetDocument;

ReadSpreadsheetDocumentAtServer(New BinaryData(FullFileName), Document);

Document.Show(Alias, FullFileName);

ElsIf Extension = ".TXT" Then

Document = New TextDocument;

Document.BeginReading(New NotifyDescription("ShowDocument", ThisForm, New Structure("Alias, Document, FullFileName", Alias, Document, FullFileName)), FullFileName);

ElsIf Extension = ".GRS" Then

Document = New GraphicalSchema;

Document.BeginReading(New NotifyDescription("ShowDocument", ThisForm, New Structure("Alias, Document, FullFileName", Alias, Document, FullFileName)), FullFileName);

Else

BeginRunningApplication(New NotifyDescription("RunApplicationNotifyHandler", ThisForm), FullFileName);

EndIf;

Return True;

EndFunction

&AtClient

Procedure RunApplicationNotifyHandler(CodeReturn, AdditionalParameters) Export

// Dummy

EndProcedure

&AtClient

Procedure ShowDocument(Params) Export

Params.Document.Show(Params.Alias, Params.FullFileName);

EndProcedure

&AtClient

Procedure ExternalFilesMountDirectory(Command)

FileDialog = New FileDialog(FileDialogMode.ChooseDirectory);

FileDialog.Show(New NotifyDescription("ExternalFilesMountDirectoryEnd", ThisForm));

EndProcedure

&AtClient

Procedure ExternalFilesMountDirectoryEnd(SelectedFiles, AdditionalParameters) Export

If SelectedFiles = Undefined Then

Return;

EndIf;

Directory = SelectedFiles[0];

NewRootItem = TreeOfFiles.GetItems().Add();

NewRootItem.Alias = Directory;

NewRootItem.FullFileName = Directory;

NewRootItem.Picture = PictureMountedDir;

AddFilesFromDirectory(NewRootItem, Directory);

Items.TreeOfFiles.Expand(NewRootItem.GetID());

Items.TreeOfFiles.CurrentRow = NewRootItem.GetID();

SaveRootItemsFromFilesTree();

EndProcedure

&AtServer

Function ConnectExternalProcessorReport(StorageAddress, IsReport)

If IsReport Then

Return ExternalReports.Connect(StorageAddress, , False);

Else

Return ExternalDataProcessors.Connect(StorageAddress, , False);

EndIf;

EndFunction

&AtServer

Function ReadSpreadsheetDocumentAtServer(BinaryData, Doc)

TFN = GetTempFileName();

BinaryData.Write(TFN);

Doc.Read(TFN);

EndFunction

&AtServer

Procedure SaveRootItemsFromFilesTree()

ArrayToSave = New Array;

RootItems = TreeOfFiles.GetItems();

For Each Row In RootItems Do

Struct = New Structure("Alias, CatalogPath, FullFileName");

FillPropertyValues(Struct, Row);

ArrayToSave.Add(Struct);

EndDo;

Try

CommonSettingsStorage.Save("СменаПользователя", "MountedDirectories", ArrayToSave);

Except

EndTry;

EndProcedure

&AtClient

Procedure TreeOfFilesOnChange(Item)

SaveRootItemsFromFilesTree();

EndProcedure

&AtClient

Procedure ExternalFilesUpdate(Command)

TreeOfFiles.GetItems().Clear();

RestoreRootItemsOfTreeOfFiles();

LoadFileTreeForRootItems();

EndProcedure

&AtClient

Procedure ExternalFilesAddToFavourite(Command)

CD = Items.TreeOfFiles.CurrentData;

If CD = Undefined Then

Message = ?(UseRu, "Сначала выберите элемент", "Select item first");

ShowUserNotification(Message, , Message,);

Return;

EndIf;

If Not CD.IsFile Then

Message = ?(UseRu, "В избранное можно добавлять только файлы, не каталоги", "Only files can be added to favourite, not directories");

ShowUserNotification(Message, , Message,);

Return;

EndIf;

NewRow = Favourite.Insert(0);

NewRow.FullFileName = CD.FullFileName;

NewRow.Name = CD.Alias;

NewRow.Picture = CD.Picture;

Items.Favourite.CurrentRow = NewRow.GetID();

EndProcedure

&НаКлиенте

Процедура CloseForm(Command)

IgnorePreventFromClosing = True;

Try

ThisForm.Close();

Except

EndTry;

КонецПроцедуры

&AtClient

Procedure RunAsUserNameStartListChoice(Item, StandardProcessing)

//RunAsUserNameStartListChoice is called in non-Taxi mode when dorp-down button is pressed

//message("RunAsUserName StartListChoice");

ListToChooseForm = GetUsersChoiceDataAtServer(RunAsUserName);

AdditionalParams = New Structure;

ShowChooseFromList(New NotifyDescription("RunAsUserHandleChoiceFromList", ThisForm, AdditionalParams), ListToChooseForm, Item);

EndProcedure

&AtClient

Procedure RunAsUserHandleChoiceFromList(SelectedValue, AdditionalParams) Export

If SelectedValue <> Undefined Then

RunAsUserName = SelectedValue.Value;

RunAsUserNameOnChange(Items.RunAsUserName);

EndIf;

EndProcedure

&AtClient

Procedure RunAsUserNameAutoComplete(Item, Text, ChoiceData, Parameters, Wait, StandardProcessing)

If Wait = True Then //it is an old version of 1C Enterprise //parameters instead of Wait and Wait instead of StandardProcessing

RunAsUserNameAutoComplete\_ForOldVersions(Item, Text, ChoiceData, Parameters, Wait)

Else

StandardProcessing = False;

ChoiceData = GetUsersChoiceDataAtServer(Text);

EndIf;

EndProcedure

&НаКлиенте

Procedure RunAsUserNameAutoComplete\_ForOldVersions(Item, Text, ChoiceData, Wait, StandardProcessing)

StandardProcessing = False;

ChoiceData = GetUsersChoiceDataAtServer(Text);

EndProcedure

&AtServerNoContext

Function GetUsersChoiceDataAtServer(Text)

ArrayOfInfoBaseUsers = InfoBaseUsers.GetUsers();

ChoiceData = New ValueList;

TrimedText = Upper(TrimAll(Text));

For Each InfoBaseUser In ArrayOfInfoBaseUsers Do

If TrimedText = "" Or Find(Upper(InfoBaseUser.Name), TrimedText) <> 0 Or Find(Upper(InfoBaseUser.FullName), TrimedText) <> 0 Then

ChoiceData.Add(InfoBaseUser.Name);

EndIf;

EndDo;

Return ChoiceData;

EndFunction

&AtClient

Procedure RunAsUserNameOnChange(Item)

UpdateAuthenticationOptionsAtServer();

EndProcedure

&AtServer

Procedure UpdateAuthenticationOptionsAtServer()

InfoBaseUser = InfoBaseUsers.FindByName(RunAsUserName);

If InfoBaseUser = Undefined Then

RunAsUserExists = False;

RunAsCurrentPasswordHash = ?(UseRu, "<Пользователь не найден>", "<User is not found>");

Else

RunAsUserExists = True;

RunAsCurrentPasswordHash = InfoBaseUser.StoredPasswordValue;

RunAsCurrentStandardAuthentication = InfoBaseUser.StandardAuthentication;

EndIf;

EndProcedure

&AtServerNoContext

Function SetAuthenticationOptionsAtServer(Username)

InfoBaseUser = InfoBaseUsers.FindByName(Username);

InfoBaseUser.Write();

EndFunction

&AtClient

Procedure RunAsChangePasswordTimeOnChange(Item)

If RunAsChangePasswordTime < RunAsChangePasswordTime\_MinValue Then

RunAsChangePasswordTime = RunAsChangePasswordTime\_MinValue;

ElsIf RunAsChangePasswordTime > RunAsChangePasswordTime\_MaxValue Then

RunAsChangePasswordTime = RunAsChangePasswordTime\_MaxValue;

EndIf;

EndProcedure

&AtClient

Procedure RunAsChangePasswordOnChange(Item)

RunAsChangePasswordOnChangeAtServer();

EndProcedure

&AtServer

Procedure RunAsChangePasswordOnChangeAtServer()

Items.RunAsTimeParams.Enabled = RunAsChangePassword;

If RunAsChangePasswordTime < RunAsChangePasswordTime\_MinValue Then

RunAsChangePasswordTime = RunAsChangePasswordTime\_MinValue;

ElsIf RunAsChangePasswordTime > RunAsChangePasswordTime\_MaxValue Then

RunAsChangePasswordTime = RunAsChangePasswordTime\_MaxValue;

EndIf;

EndProcedure

&AtClient

Procedure RunAsRun(Command)

#If Not WebClient Then

UpdateAuthenticationOptionsAtServer();

If Not RunAsUserExists Or IsBlankString(RunAsUserName) Then

Message = ?(UseRu, "ПОЛЬЗОВАТЕЛЬ НЕ НАЙДЕН!", "USER WAS NOT FOUND!");

ShowUserNotification(Message, , Message, PictureLib.InputOnBasis);

Return;

EndIf;

//ParamString = "DESIGNER /RunModeOrdinaryApplication " + InfoBaseConnectionString() + "/N""" + RunAsUserName + """";

//ParamString = "ENTERPRISE /RunModeOrdinaryApplication " + InfoBaseConnectionString() + "/N""" + RunAsUserName + """";

ParamString = "ENTERPRISE /RunModeManagedApplication /Debug " + InfoBaseConnectionString() + " /N""" + RunAsUserName + """";

If RunAsChangePassword Then

If RunAsChangePasswordTime < RunAsChangePasswordTime\_MinValue Then

RunAsChangePasswordTime = RunAsChangePasswordTime\_MinValue;

ElsIf RunAsChangePasswordTime > RunAsChangePasswordTime\_MaxValue Then

RunAsChangePasswordTime = RunAsChangePasswordTime\_MaxValue;

EndIf;

Try

If Items.ForClipboardData.Document.parentWindow.ClipboardData.SetData("Text", RunAsCurrentPasswordHash) Then

Message = ?(UseRu, "Текущий хэш пароля был скопирован в буфер обмена!", "Current password hash has been copied to the clipboard!");

ShowUserNotification(Message, , Message, PictureLib.InputOnBasis);

EndIf;

Except

EndTry;

ParamString = ParamString;

SetAuthenticationOptionsAtServer(RunAsUserName);

Items.Settings\_Language.Enabled = False;

Items.RunAsUserName.Enabled = False;

Items.RunAsRun.Enabled = False;

Items.RunAsRun.Title = ?(UseRu, "Ожидание восстановления пароля", "Awaiting password to be restored");

RunAs\_PreventCloseWhileAwaitingPasswordToBeRestored = True;

RunAsSecondsLeftToRestorePassword = RunAsChangePasswordTime - 2; // adjustment = 2

Items.RunAsSecondsLeftToRestorePassword.Visible = True;

AttachIdleHandler("RestorePasswordHash", RunAsChangePasswordTime, True);

AttachIdleHandler("RestorePasswordCountdown", 1, False);

EndIf;

RunSystem(ParamString + ?(IsBlankString(RunAsAdditionalParams), "", " " + RunAsAdditionalParams));

ПрекратитьРаботуСистемы(Ложь,)

#EndIf

EndProcedure

&AtClient

Procedure RestorePasswordCountdown() Export

If Not RunAs\_PreventCloseWhileAwaitingPasswordToBeRestored Then

Return;

EndIf;

If RunAsSecondsLeftToRestorePassword > 0 Then

RunAsSecondsLeftToRestorePassword = RunAsSecondsLeftToRestorePassword - 1;

EndIf;

EndProcedure

&AtClient

Procedure RestorePasswordHash() Export

Items.RunAsRun.Title = ?(UseRu, "Запустить приложение", "Run application");

Items.Settings\_Language.Enabled = True;

Items.RunAsUserName.Enabled = True;

Items.RunAsRun.Enabled = True;

RunAs\_PreventCloseWhileAwaitingPasswordToBeRestored = False;

Items.RunAsSecondsLeftToRestorePassword.Visible = False;

DetachIdleHandler("RestorePasswordCountdown");

SetAuthenticationOptionsAtServer(RunAsUserName);

Message = ?(UseRu, "Пароль был восстановлен!", "Password has been restored!");

ShowUserNotification(Message, , Message, PictureLib.InputOnBasis);

EndProcedure

&AtServer

Procedure OnLoadDataFromSettingsAtServer(Settings)

RunAsChangePasswordOnChangeAtServer();

UpdateAuthenticationOptionsAtServer();

EndProcedure

&AtClient

Procedure ClearHistory(Command)

History.Clear();

EndProcedure

&AtClient

Procedure Settings\_ChangeCaptionOfTheMainWindowOnChange(Item)

Items.GroupChangeCaptionOfTheMainWindow.Enabled = Settings\_ChangeCaptionOfTheMainWindow;

If Settings\_ChangeCaptionOfTheMainWindow Then

Settings\_ChangeCaptionOfTheMainWindowOnChangeAtServer();

SetApplicationCaption(Settings\_CaptionOfTheMainWindow);

EndIf;

SettingsItemChanged(Item);

EndProcedure

&AtServer

Procedure Settings\_ChangeCaptionOfTheMainWindowOnChangeAtServer()

If Settings\_ChangeCaptionOfTheMainWindow Then

DBName = "";

If Settings\_ShowDatabaseNameInCaption Then

ConnectionString = InfoBaseConnectionString();

DBNamePosition = Find(ConnectionString,"Ref");

If DBNamePosition = Undefined Or DBNamePosition = 0 Тогда

DBName = Mid(ConnectionString,7,StrLen(ConnectionString)-8);

Else

DBName = Right(ConnectionString,StrLen(ConnectionString)-DBNamePosition-4);

DBName = Left(DBName,StrLen(DBName)-2);

EndIf;

EndIf;

Settings\_CaptionOfTheMainWindow = ?(Settings\_ShowDatabaseNameInCaption, DBName, "") + ?(Settings\_ShowCurrentUsernameInCaption, ?(Settings\_ShowDatabaseNameInCaption, " - ", "") + InfoBaseUsers.CurrentUser().Name, "") + ?(IsBlankString(Settings\_TextToAddToCaption), "", " - " + Settings\_TextToAddToCaption);

EndIf;

EndProcedure

&AtClient

Procedure Settings\_ApplyVisabilityFlagsOnClient(Item)

ApplyVisabilityFlags();

SettingsItemChanged(Item);

EndProcedure

&AtClient

Procedure BeforeClose(Cancel, StandardProcessing)

If RunAs\_PreventCloseWhileAwaitingPasswordToBeRestored Then

Message = ?(UseRu, "Ожидание восстановления пароля пользователя. Форма не может быть сейчас закрыта." , "Awaiting for user's password to be restored. Form cannot be closed now.");

ShowUserNotification(Message, , Message, PictureLib.Delete);

Cancel = True;

Return;

EndIf;

If Settings\_AskBeforeClose And Not IgnorePreventFromClosing Then

Cancel = True;

QueryText = ?(UseRu, "Закрыть рабочий стол администратора?", "Do you want to close admin's workspace?");

Buttons = QuestionDialogMode.YesNo;

ShowQueryBox(New NotifyDescription("AskBeforeCloseHandleSelection", ThisForm), QueryText, Buttons);

EndIf;

EndProcedure

&AtClient

Procedure AskBeforeCloseHandleSelection(SelectedValue, AdditionalParams = Undefined) Export

If SelectedValue = DialogReturnCode.Yes Then

IgnorePreventFromClosing = True;

Try

ThisForm.Close();

Except

EndTry;

EndIf;

EndProcedure

&AtClient

Procedure Settings\_LanguageOnChange(Item)

SettingsItemChanged(Item);

OpenForm(FormName, , , New UUID);

IgnorePreventFromClosing = True;

Try

ThisForm.Close();

Except

Message = ?(UseRu,"Изменения вступят в силу при следующем запуске", "Changes will be applied on next startup");

ShowUserNotification(Message, , Message);

EndTry;

EndProcedure

&AtClient

Procedure InitializeTranslateMaps()

CharsMapEnToRu = New Map;

CharsMapRuToEn = New Map;

CharsMapEnToRu.Insert("`","ё");

CharsMapEnToRu.Insert("~","ё");

CharsMapEnToRu.Insert("[","х");

CharsMapEnToRu.Insert("{","х");

CharsMapEnToRu.Insert("]","ъ");

CharsMapEnToRu.Insert("}","ъ");

CharsMapEnToRu.Insert(",","б");

CharsMapEnToRu.Insert("<","б");

CharsMapEnToRu.Insert(".","ю");

CharsMapEnToRu.Insert(">","ю");

CharsMapEnToRu.Insert(":","ж");

CharsMapEnToRu.Insert(";","ж");

CharsMapEnToRu.Insert("'","э");

CharsMapEnToRu.Insert("""","э");

AddToMaps("q","й");

AddToMaps("w","ц");

AddToMaps("e","у");

AddToMaps("r","к");

AddToMaps("t","е");

AddToMaps("y","н");

AddToMaps("u","г");

AddToMaps("i","ш");

AddToMaps("o","щ");

AddToMaps("p","з");

AddToMaps("a","ф");

AddToMaps("s","ы");

AddToMaps("d","в");

AddToMaps("f","а");

AddToMaps("g","п");

AddToMaps("h","р");

AddToMaps("j","о");

AddToMaps("k","л");

AddToMaps("l","д");

AddToMaps("z","я");

AddToMaps("x","ч");

AddToMaps("c","с");

AddToMaps("v","м");

AddToMaps("b","и");

AddToMaps("n","т");

AddToMaps("m","ь");

EndProcedure

&AtClient

Procedure AddToMaps(CharEn, CharRu);

CharsMapEnToRu.Insert(CharEn, CharRu);

CharsMapRuToEn.Insert(CharRu, CharEn);

EndProcedure

&AtClient

Function GetMapedString(String, ToEnglish);

Map = ?(ToEnglish, CharsMapRuToEn, CharsMapEnToRu);

OutputString = "";

Length = StrLen(String);

For K = 1 To Length Do

Char = Mid(String,K,1);

OutputChar = Map.Get(Char);

If OutputChar = Undefined Then

OutputChar = Char;

EndIf;

OutputString = OutputString + OutputChar;

EndDo;

Return OutputString;

EndFunction

&AtClient

Procedure SearchStringAutoComplete(Item, Text, ChoiceData, Parameters, Wait, StandardProcessing)

Text = TrimAll(Text);

SearchStringQueryToRemember = Text; //if value will be choosed we'll place this string into SearchStringRememberedQueries

If Wait = True Then //it is an old version of 1C Enterprise //parameters instead of Wait and Wait instead of StandardProcessing

SearchStringAutoComplete\_ForOldVersions(Item, Text, ChoiceData, Parameters, Wait)

Else

StandardProcessing = False;

ChoiceData = GetChoiceDataForSearchString(Text);

EndIf;

EndProcedure

&AtClient

Procedure SearchStringAutoComplete\_ForOldVersions(Item, Text, ChoiceData, Wait, StandardProcessing)

StandardProcessing = False;

ChoiceData = GetChoiceDataForSearchString(Text);

EndProcedure

&AtClient

Function GetChoiceDataForSearchString(Text)

ChoiceDataRaw = New ValueList;

If IsBlankString(Text) Then

If IsBlankString(SearchString) Then

Return ChoiceDataRaw;

Else

Text = SearchString

EndIf;

EndIf;

Original = Lower(Text);

OriginalToRu = GetMapedString(Original, False);

OriginalToEn = GetMapedString(Original, True);

LengthOriginal = StrLen(Original);

LengthOriginalToRu = StrLen(OriginalToRu);

LengthOriginalToEn = StrLen(OriginalToEn);

//Message("1 - " + Original);

//Message("2 - " + OriginalToRu);

//Message("3 - " + OriginalToEn);

For Each MDClassRow In MetadataClasses Do

Picture = MDClassRow.Picture;

MDAttribure = ThisForm[MDClassRow.NameEn];

For Each MDRow In MDAttribure Do

Name = MDRow.Name;

Synonym = MDRow.Synonym;

LowerName = Lower(Name);

LowerSynonym = Lower(Synonym);

If

Find(LowerName, Original) <> 0 Or

Find(LowerSynonym, Original) <> 0 Or

Find(LowerName, OriginalToRu) <> 0 Or

Find(LowerSynonym, OriginalToRu) <> 0 Or

Find(LowerName, OriginalToEn) <> 0 Or

Find(LowerSynonym, OriginalToEn) <> 0

Then

Order = 9;

If Left(LowerName, LengthOriginal) = Original Or Left(LowerSynonym, LengthOriginal) = Original Then //if name or synonym begins with original text

Order = 1;

ElsIf Left(LowerName, LengthOriginalToEn) = OriginalToEn Or Left(LowerSynonym, LengthOriginalToEn) = OriginalToEn Then //if name of synonym beging with text mapped in EN

Order = 2;

ElsIf Left(LowerName, LengthOriginalToRu) = OriginalToRu Or Left(LowerSynonym, LengthOriginalToRu) = OriginalToRu Then //if name of synonym beging with text mapped in RU

Order = 3;

EndIf;

ChoiceDataRaw.Add(?(UseRu, MDClassRow.NameRu, MDClassRow.NameEn) + "."+ Name, String(Order));

EndIf;

EndDo;

EndDo;

SearchTreeOfFiles(TreeOfFiles.GetItems(), ChoiceDataRaw, Original, OriginalToRu, OriginalToEn, LengthOriginal, LengthOriginalToRu, LengthOriginalToEn);

ChoiceDataRaw.SortByPresentation(SortDirection.Asc);

ChoiceData = New ValueList;

K = 0;

For Each Item In ChoiceDataRaw Do

ChoiceData.Add(Item.Value);

K = K + 1;

If K = 40 Then

Break;

EndIf;

EndDo;

Return ChoiceData;

EndFunction

&AtClient

Procedure SearchTreeOfFiles(TreeItems, ChoiceDataRaw, Original, OriginalToRu, OriginalToEn, LengthOriginal, LengthOriginalToRu, LengthOriginalToEn)

For Each Item In TreeItems Do

If Not Item.IsFile Then

SearchTreeOfFiles(Item.GetItems(), ChoiceDataRaw, Original, OriginalToRu, OriginalToEn, LengthOriginal, LengthOriginalToRu, LengthOriginalToEn);

Else

Name = Item.Alias;

LowerName = Lower(Name);

If

Find(LowerName, Original) <> 0 Or

Find(LowerName, OriginalToRu) <> 0 Or

Find(LowerName, OriginalToEn) <> 0

Then

Order = 9;

If Left(LowerName, LengthOriginal) = Original Then

Order = 1;

ElsIf Left(LowerName, LengthOriginalToEn) = OriginalToEn Then

Order = 2;

ElsIf Left(LowerName, LengthOriginalToRu) = OriginalToRu Then

Order = 3;

EndIf;

ChoiceDataRaw.Add(?(UseRu, "<Файлы>", "<Files>") + "."+ Name +

" "

+"<"+String(Item.GetID())+">", String(Order));

EndIf;

EndIf;

EndDo;

EndProcedure

&AtClient

Procedure SearchStringOnChange(Item)

ErrorMessage = ?(UseRu,"Пожалуйста, выберите значение из выпадающего списка", "Please select value from the drop-down list");

MessageSuccess = ?(UseRu,"Объект найден и выделен в списке", "Object is found and highlighted in the list");

PeriodPosition = Find(SearchString, ".");

If PeriodPosition = 0 Then

ShowUserNotification(ErrorMessage, , ErrorMessage);

Return;

EndIf;

ClassName = Left(SearchString, PeriodPosition -1);

ObjectName = Mid(SearchString, PeriodPosition +1);

If ClassName = ?(UseRu, "<Файлы>", "<Files>") Then

//extracting row ID

FirstPos = Find(ObjectName, "<");

LastPos = Find(ObjectName, ">");

If FirstPos = 0 Or LastPos = 0 Or FirstPos > LastPos Then

ShowUserNotification(ErrorMessage, , ErrorMessage);

Return;

EndIf;

Try

ID = Number(Mid(ObjectName, FirstPos+1, LastPos - FirstPos - 1));

Except

ShowUserNotification(ErrorMessage, , ErrorMessage);

Return;

EndTry;

SearchString = TrimAll(Left(ObjectName, FirstPos -1));

Items.FormPages.CurrentPage = Items.ExternalFiles;

ThisForm.CurrentItem = Items.TreeOfFiles;

Items.TreeOfFiles.CurrentRow = ID;

Else

ClassRows = MetadataClasses.FindRows(New Structure("NameEn", ClassName));

If ClassRows.Count() = 0 Then

ClassRows = MetadataClasses.FindRows(New Structure("NameRu", ClassName));

EndIf;

If ClassRows.Count() = 0 Or ObjectName = "" Then

ShowUserNotification(ErrorMessage, , ErrorMessage);

Return;

EndIf;

ClassRow = ClassRows[0];

ObjectsTable = ThisForm[ClassRow.NameEn];

ObjectRows = ObjectsTable.FindRows(New Structure("Name", ObjectName));

If ObjectRows.Count() = 0 Then

ShowUserNotification(ErrorMessage, , ErrorMessage);

Return;

EndIf;

ObjectRow = ObjectRows[0];

SearchString = ObjectName;

Items.MetadataClasses.CurrentRow = ClassRow.GetID();

Items.FormPages.CurrentPage = Items.DatabaseObjects;

Items.PagesObjects.CurrentPage = Items["Page"+ClassRow.NameEn];

Items[ClassRow.NameEn].CurrentRow = ObjectRow.GetID();

ThisForm.CurrentItem = Items[ClassRow.NameEn];

EndIf;

ShowUserNotification(MessageSuccess, , MessageSuccess);

RememberLastSearchQuery();

EndProcedure

&AtClient

Procedure RememberLastSearchQuery()

If IsBlankString(SearchStringQueryToRemember) Then

Return;

EndIf;

Item = SearchStringRememberedQueries.FindByValue(SearchStringQueryToRemember);

If Item <> Undefined Then

SearchStringRememberedQueries.Delete(Item);

EndIf;

SearchStringRememberedQueries.Insert(0, SearchStringQueryToRemember);

Count = SearchStringRememberedQueries.Count();

If Count > 10 Then

ItemsToRemove = New Array;

For K = 10 To Count-1 Do

ItemsToRemove.Add(SearchStringRememberedQueries[K]);

EndDo;

For Each Item In ItemsToRemove Do

SearchStringRememberedQueries.Delete(Item);

EndDo;

EndIf;

EndProcedure

&AtClient

Procedure SearchStringStartListChoice(Item, StandardProcessing)

EndProcedure

&AtClient

Procedure SearchStringStartListChoiceHandleSelection(SelectedValue, AdditionalParams) Export

If SelectedValue <> Undefined Then

SearchString = SelectedValue.Value;

SearchStringOnChange(Items.SearchString);

EndIf;

EndProcedure

&AtClient

Procedure SearchStringOpening(Item, StandardProcessing)

StandardProcessing = False;

If Items.SearchString.OpenButton Then // Button is visible

List = GetChoiceDataForSearchString(TrimAll(SearchString));

ShowChooseFromList(New NotifyDescription("SearchStringStartListChoiceHandleSelection", ThisForm), List, Item);

EndIf;

EndProcedure

&AtClient

Procedure SearchStringStartChoice(Item, ChoiceData, StandardProcessing)

StandardProcessing = False;

AdditionalParams = New Structure;

ShowChooseFromList(New NotifyDescription("SearchStringStartChoiceHandleSelection", ThisForm), SearchStringRememberedQueries, Item);

EndProcedure

&AtClient

Procedure SearchStringStartChoiceHandleSelection(SelectedValue, AdditionalParams) Export

If SelectedValue <> Undefined Then

SearchString = SelectedValue.Value;

SearchStringOpening(Items.SearchString, False);

EndIf;

EndProcedure

&AtClient

Procedure ProjectLinkClick(Item)

BeginRunningApplication( New NotifyDescription("RunApplicationNotifyHandler", ThisForm) ,Mid(Items.ProjectLink.Title, Find(Items.ProjectLink.Title, "http://")));

EndProcedure

&AtClient

Procedure SettingsItemChanged(Item)

SaveSettings();

EndProcedure