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 14](#_Toc122583446)

[3 ЗАКЛЮЧЕНИЕ 17](#_Toc122583447)

[3.1 Приложение A. Листинг окон программы 17](#_Toc122583448)

[3.2 Приложение В. Листинг кода программы 30](#_Toc122583449)

[3.2.1 Модуль общей формы МенюПреподавателя: 30](#_Toc122583450)

[3.2.2 Модуль общей формы МенюУченика: 32](#_Toc122583451)

[3.2.3 Модуль общей формы МенюАдмина 33](#_Toc122583452)

[3.2.4 Модуль формы элемента справочника Теория 35](#_Toc122583453)

[3.2.5 Запрос схемы компоновки данных отчета Расписание 37](#_Toc122583454)

[3.2.6 Запрос схемы компоновки данных отчета Успеваемость 38](#_Toc122583455)

[3.2.7 Модуль формы обработки Тест 39](#_Toc122583456)

[3.2.8 Модуль формы MainFrom обработки СменаПользователя: 46](#_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.

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

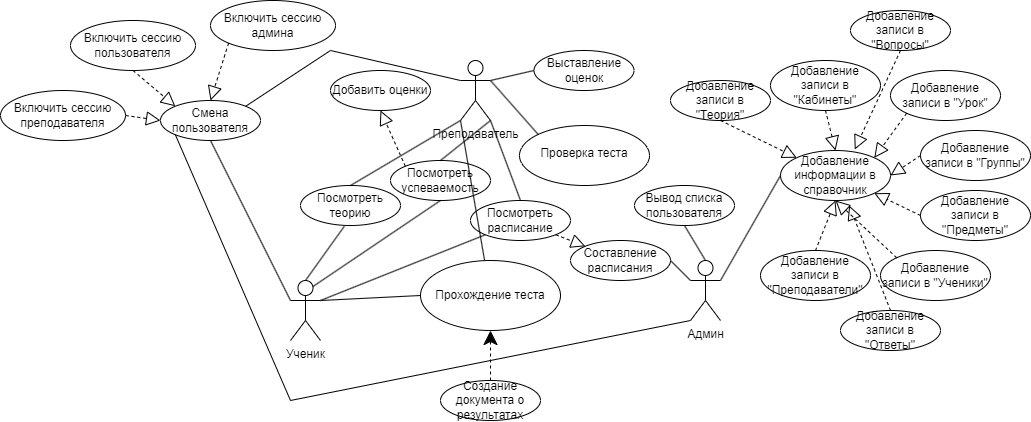
**СписокПользователей** –

1. Выводит список пользователей системы в меню админа

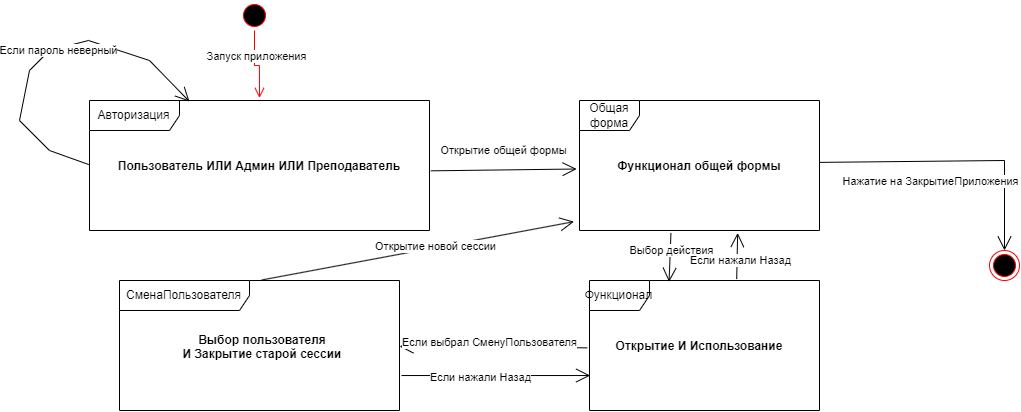
**Добавление объектов** –

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

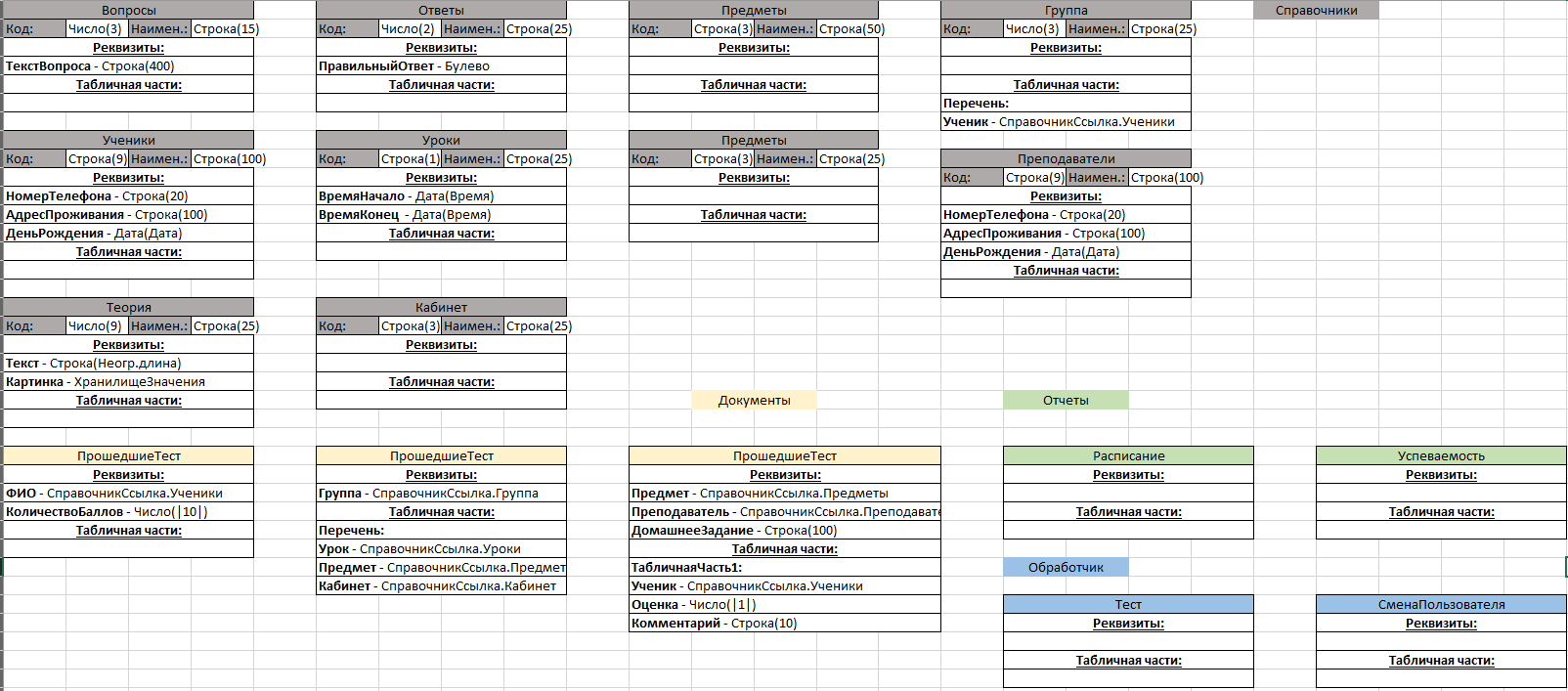
**Диаграмма состояний** - это тип диаграммы, используемый в информатике и смежных областях для описания поведения систем. Диаграммы состояний требуют, чтобы описываемая система состояла из конечного числа состояний; иногда это действительно так, в то время как в других случаях это разумная абстракция. 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

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.

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** | Ученик, админ или преподаватель авторизуется в систему. |
| **Фактический результат** | Авторизация прошла успешно в не зависимости, кого выбирали. |
| **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 |

Test case №3:

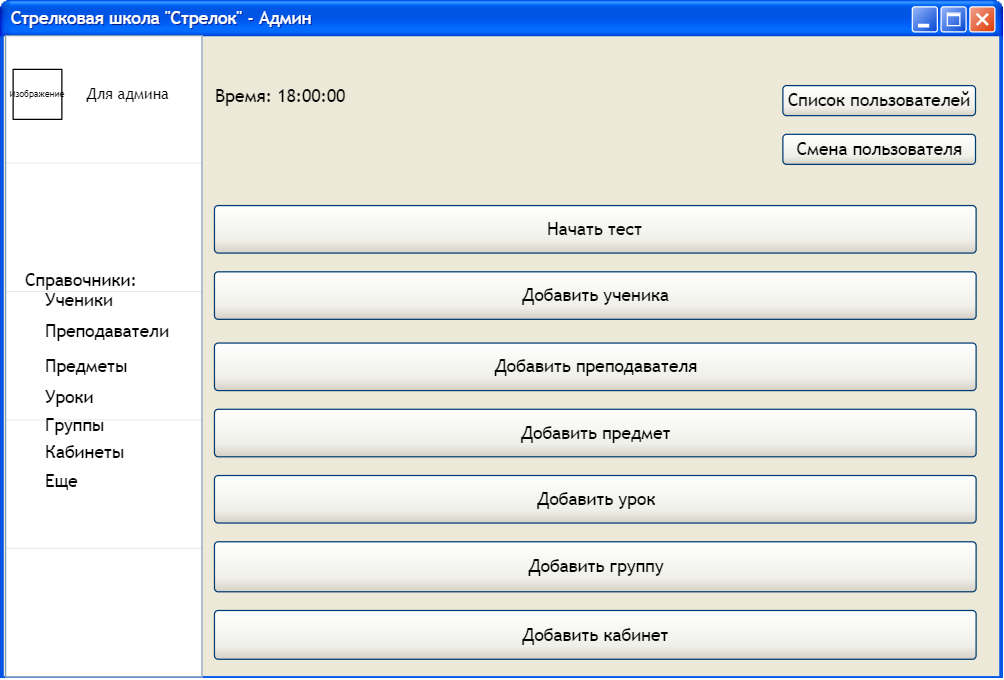
|  |  |
| --- | --- |
| **Test Case №** | ТС\_3 |
| **Приоритет теста** | Среднее |
| **Название тестирования/Имя** | Проверка функции «СменаПользователя». |
| **Резюме испытания** | Пользователь выбирает из списка пользователей, чтобы загрузить новую сессию за выбранного пользователя. |
| **Шаги тестирования** | * 1. Запуск программы под любым пользователем.   2. Нажимаем кнопку «Смена пользователя», после перехода на другую страницу в поле «Имя пользователя» выбираем пользователя.   3. Нажимаем кнопку «Загрузить новую сессию». |
| **Ожидаемый результат** | Запуститься новая сессия с выбранным пользователем и закроется старая сессия. |
| **Фактический результат** | Закрылось текущее окно и открылось новое окно с тем пользователем, которого выбирали. |
| **Предпосылки** |  |
| **Статус (Зачет/Незачет)** | Зачет |
| **Комментарии** |  |

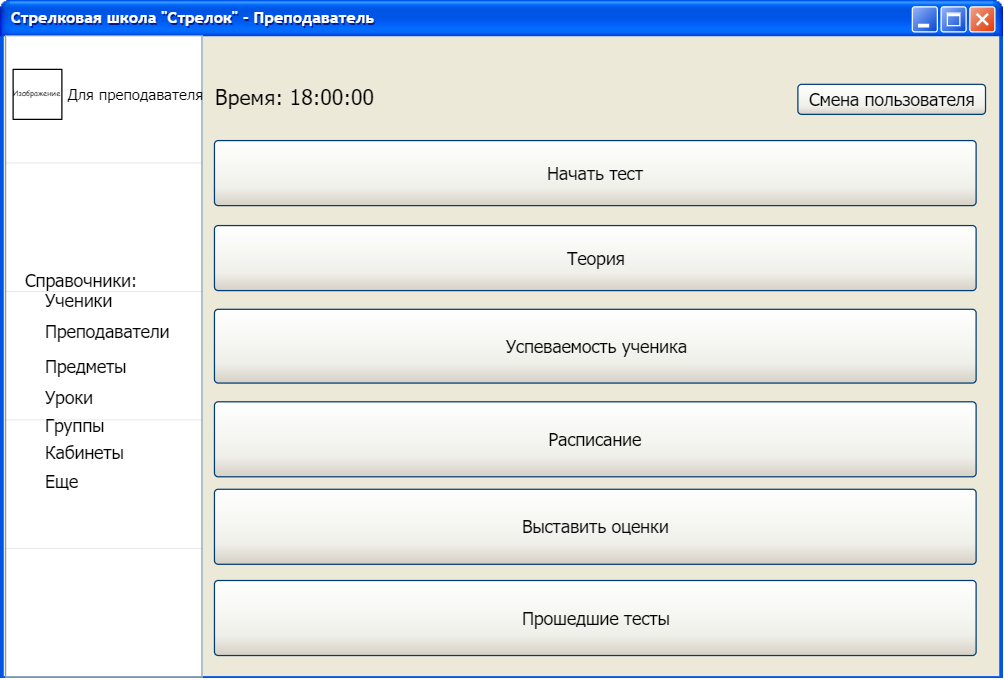
Test case №4:

|  |  |
| --- | --- |
| **Test Case №** | ТС\_4 |
| **Приоритет теста** | Среднее |
| **Название тестирования/Имя** | Проверка функции «Тест» |
| **Резюме испытания** | Пользователь может пройти тест. |
| **Шаги тестирования** | 1. Запуск программы либо под пользователем «Ученик», либо «Преподаватель» 2. Нажимаем кнопку «Начать тест», после перехода на другую страницу в поле «ФИО ученика» выбираем ученика из списка. 3. Нажимаем кнопку «Начать тест». |
| **Ожидаемый результат** | Откроется страница с вопросами, ответив на них в конце должен появиться результат с количеством правильных ответов и покажется сообщении о создании документа о прохождении теста данным учеником. |
| **Фактический результат** | Открылась страница с вопросами, после ответа на них появился результат и сообщение про документ. |
| **Предпосылки** | Чтобы проверить создаться документ необходимо зайти в документ «Прошедшие тест» |
| **Статус (Зачет/Незачет)** | Зачет |
| **Комментарии** |  |

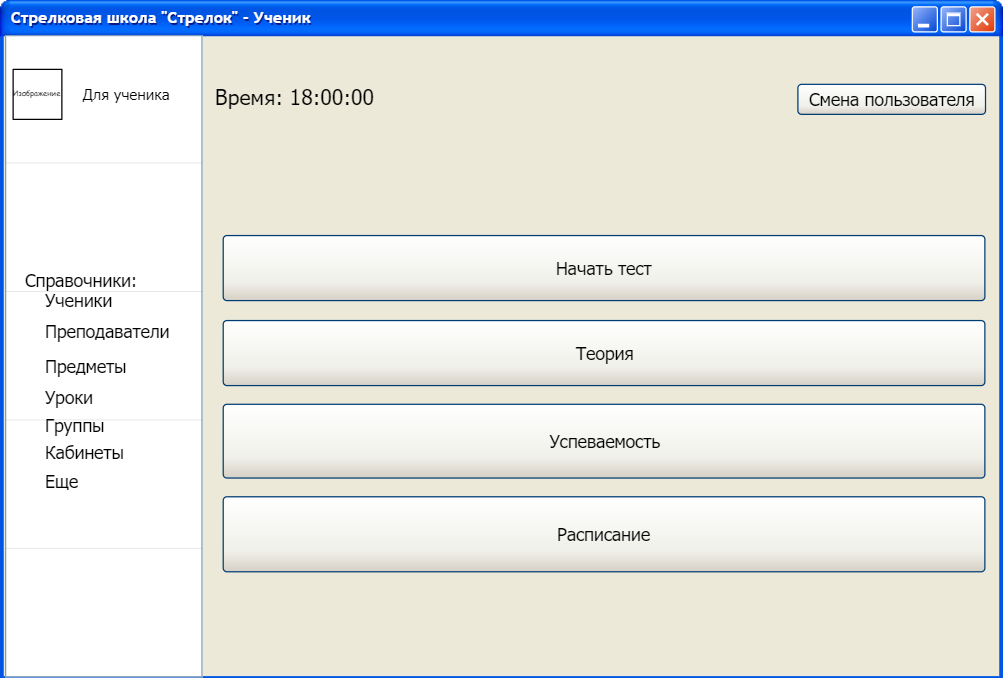
* 1. Прототипы основных интерфейсов

**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



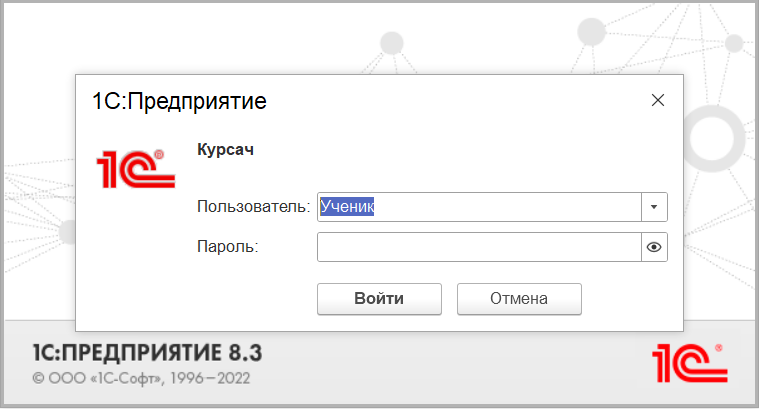
Прототип окна преподавателя

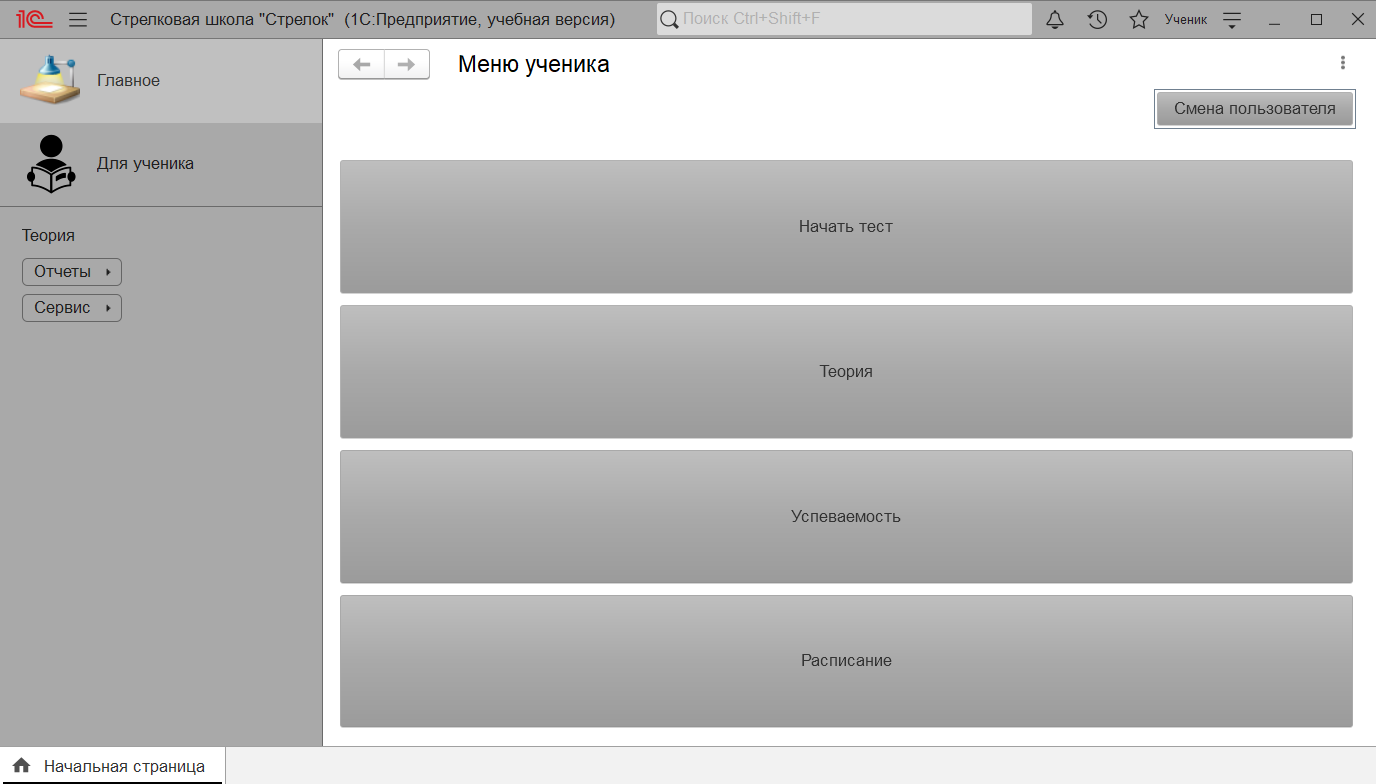


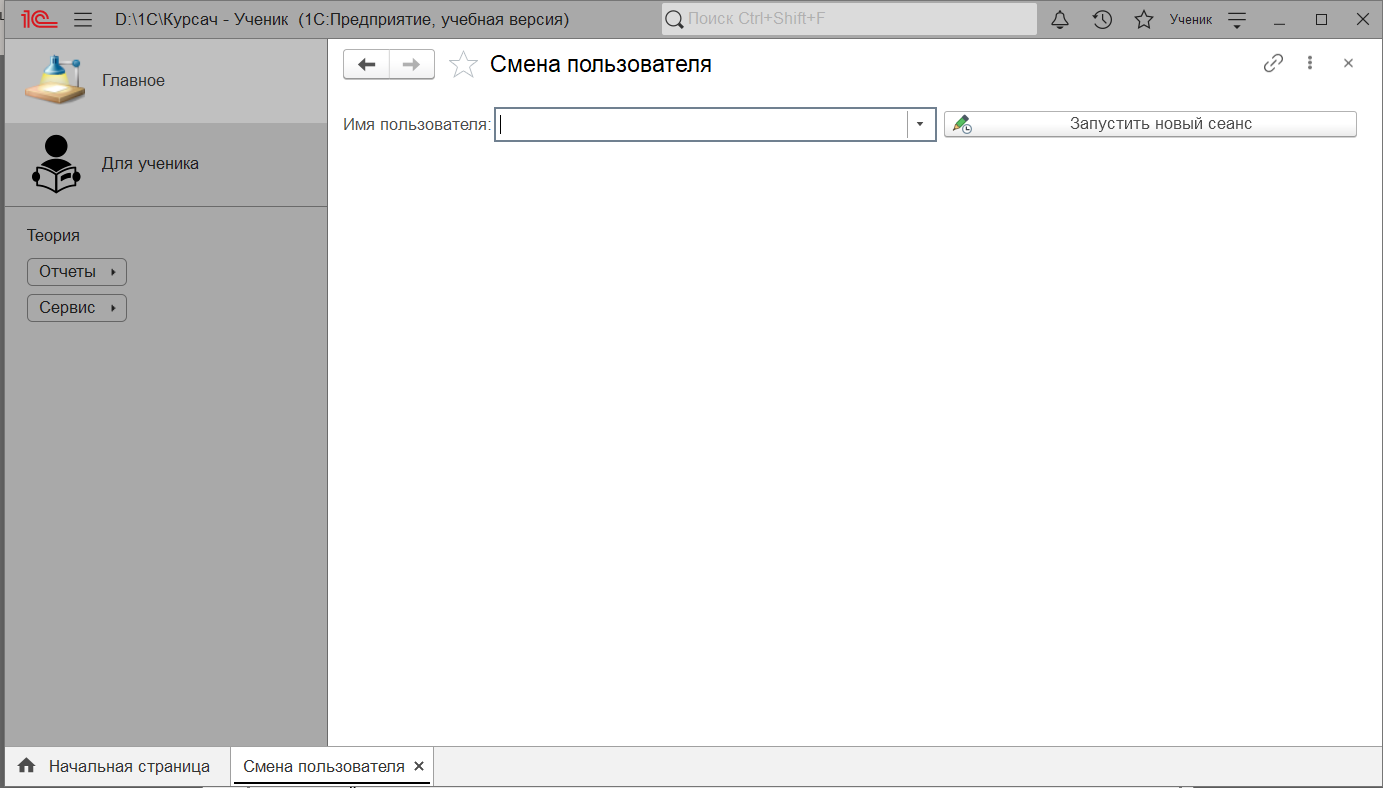
Прототип окна ученика

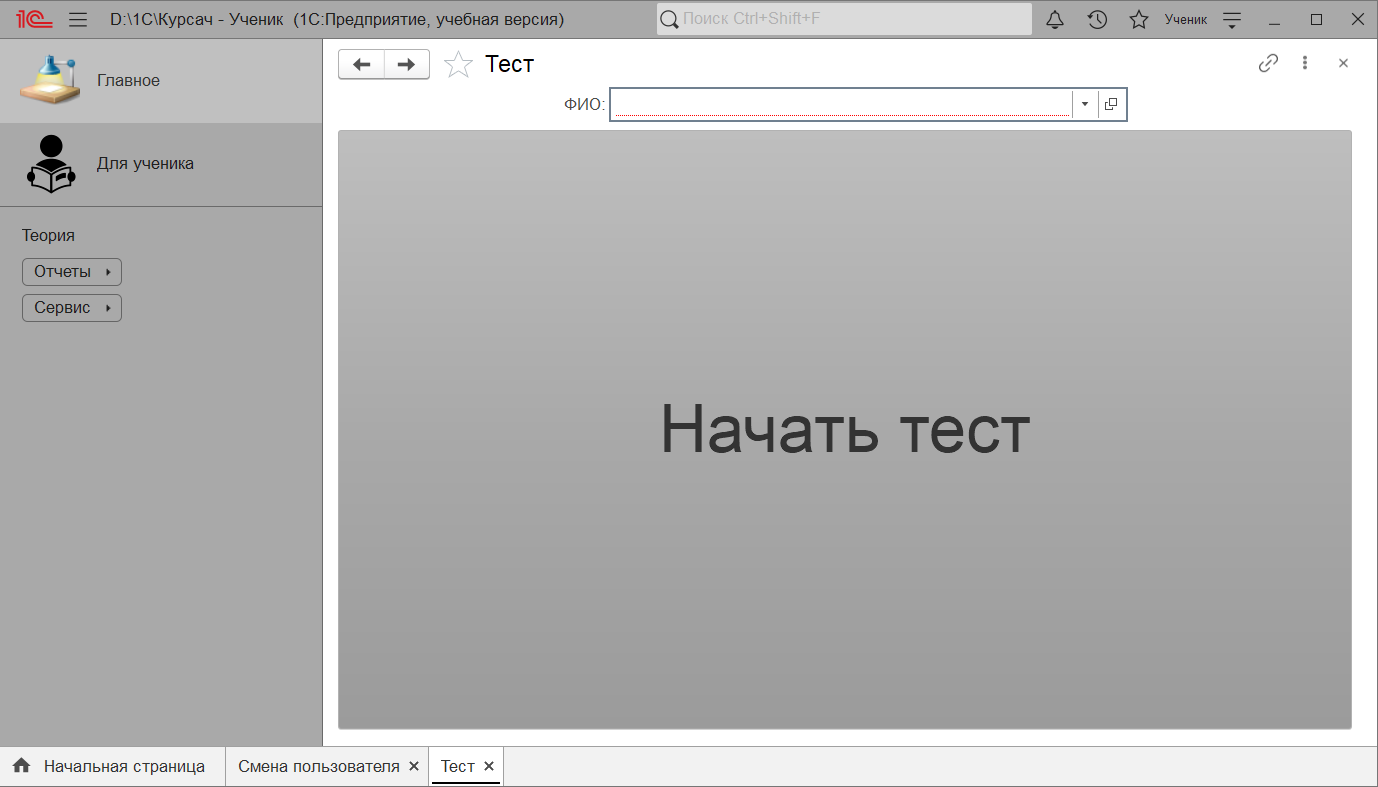
ЗАКЛЮЧЕНИЕ

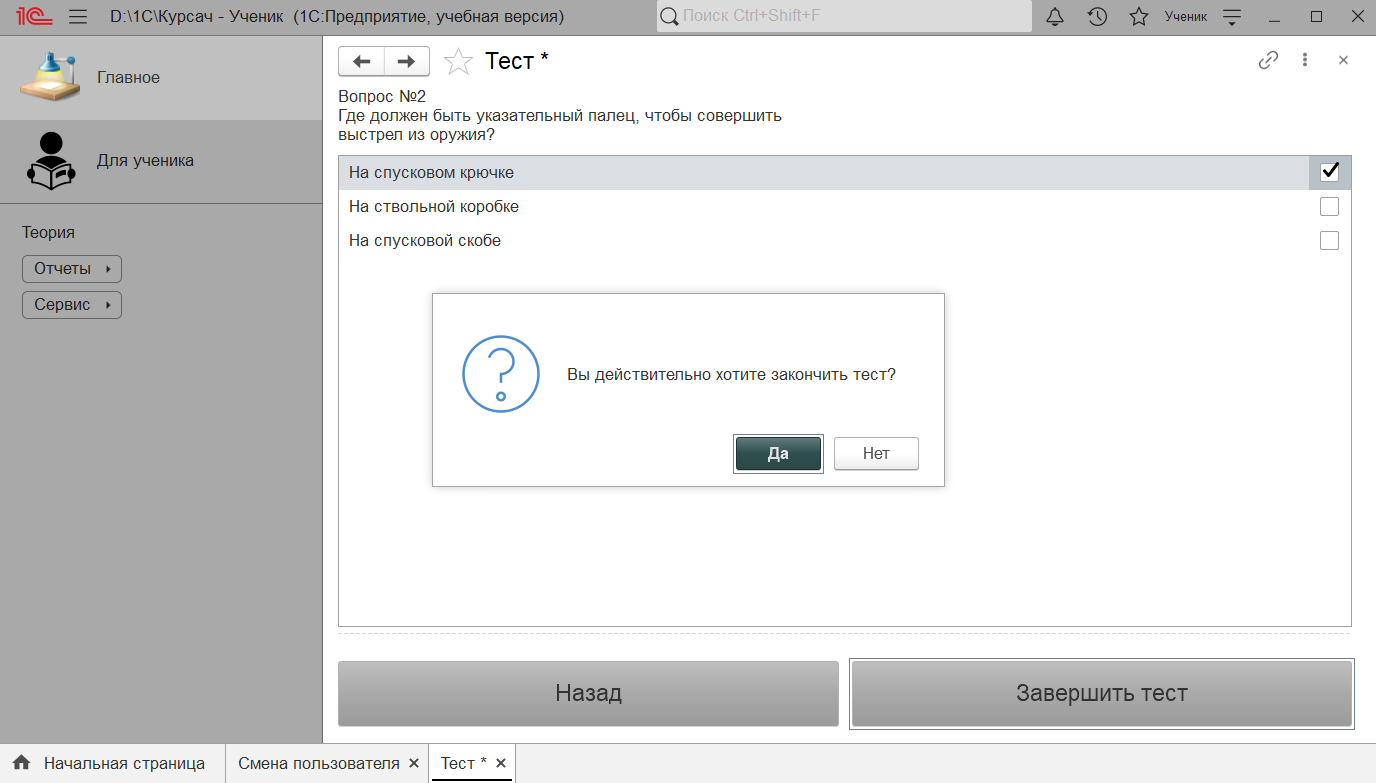
* 1. Приложение A. Листинг окон программы

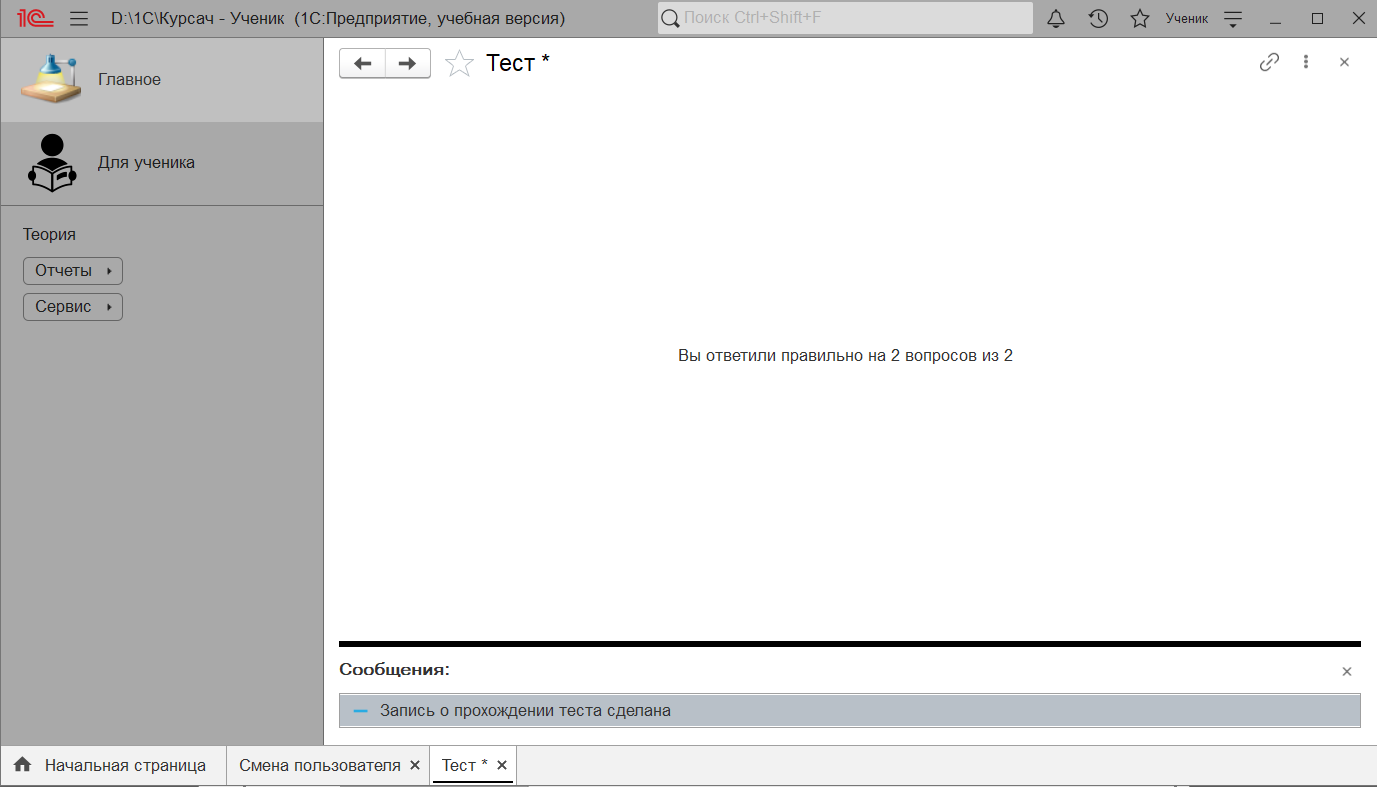


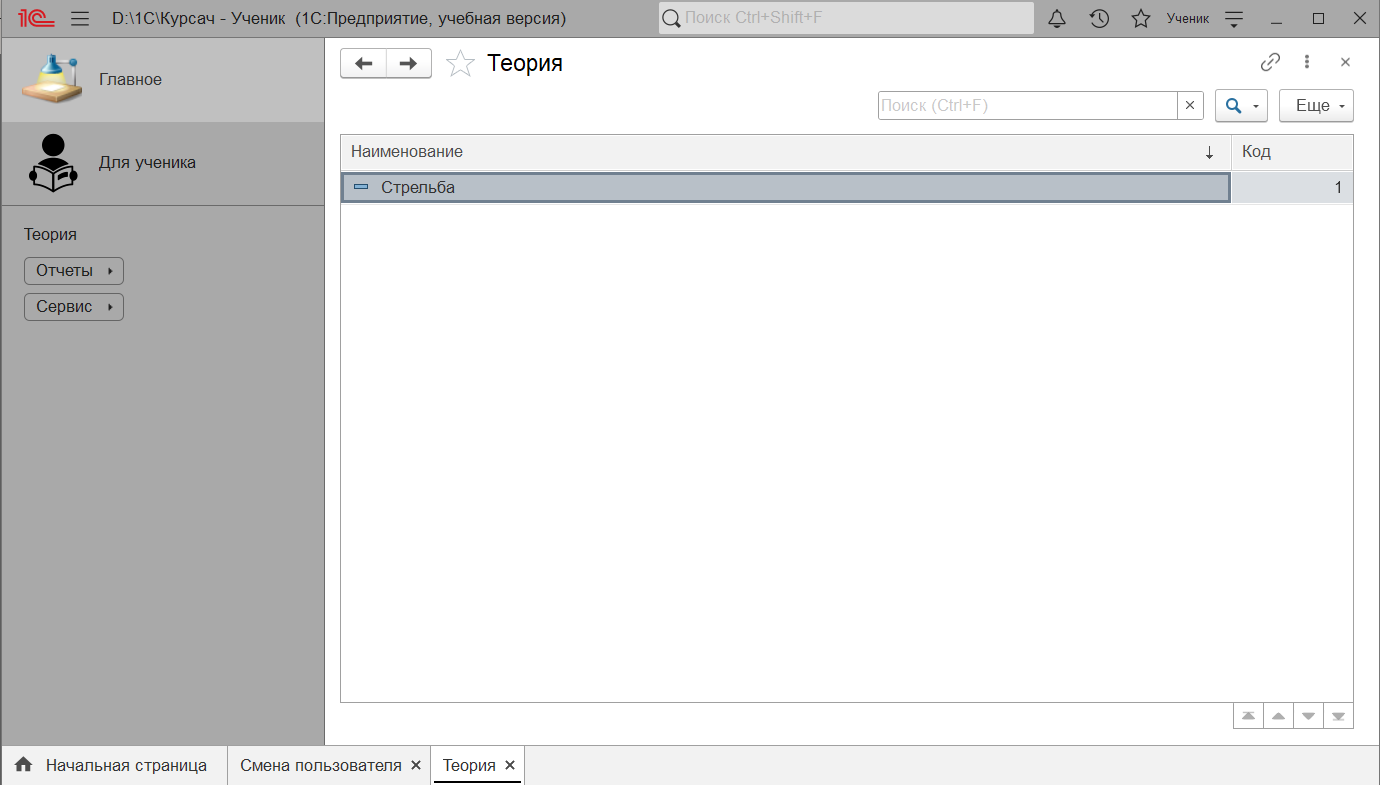


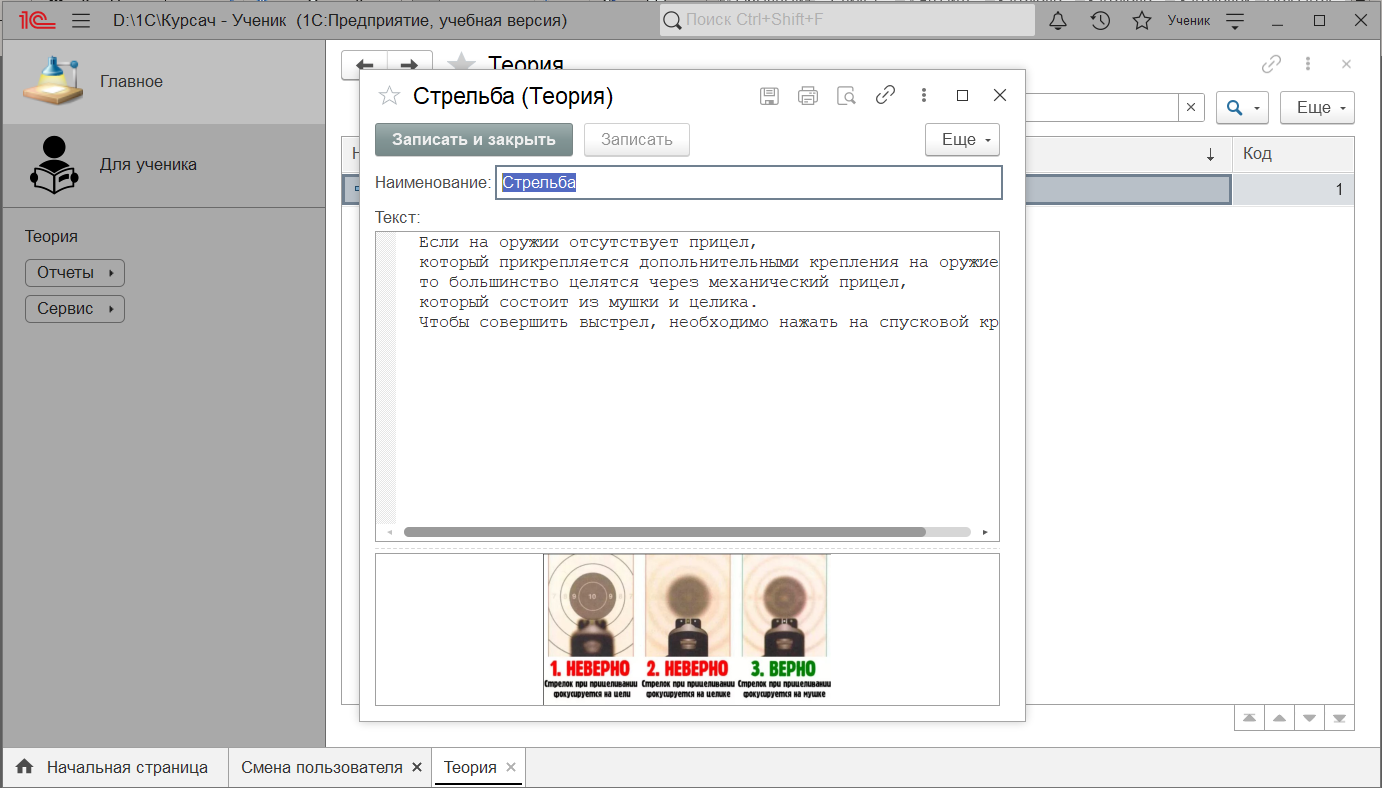


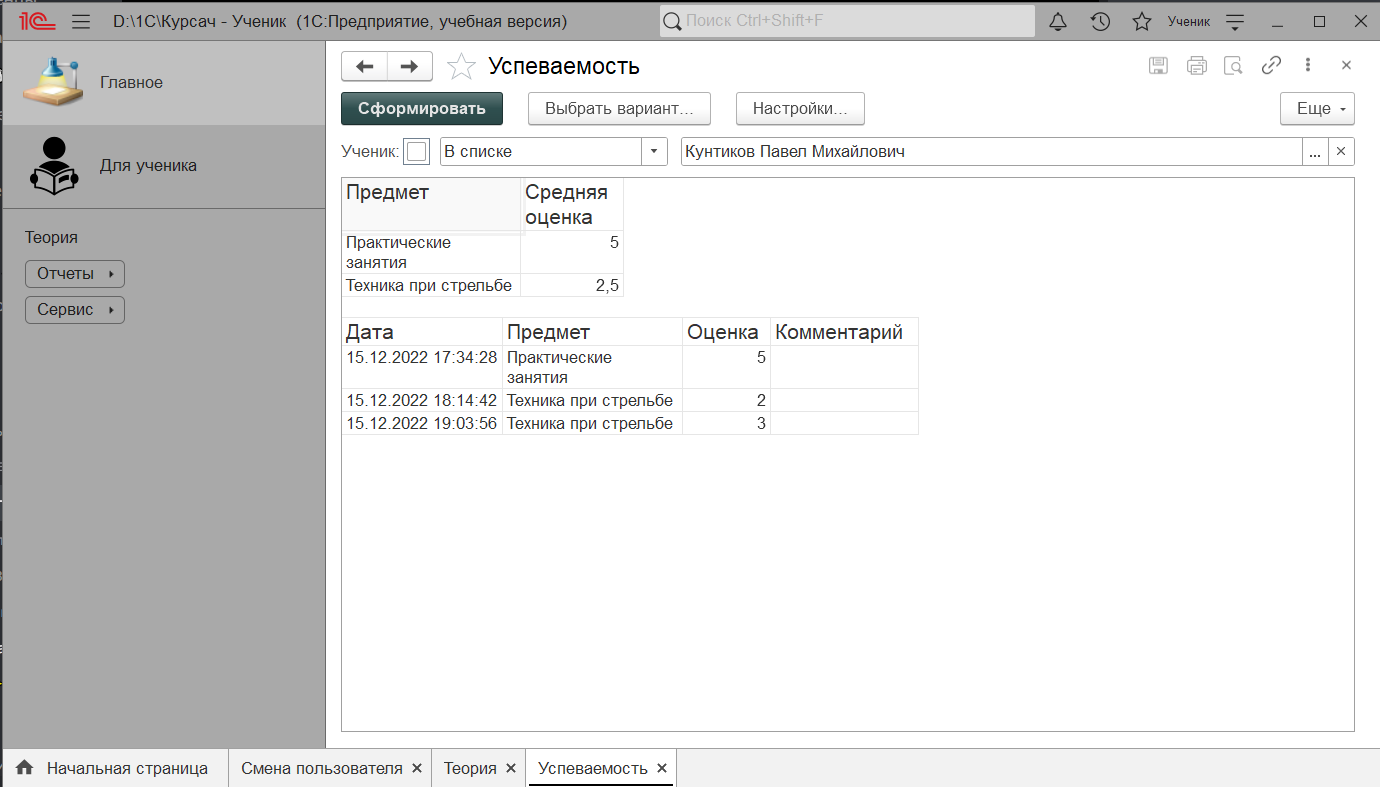


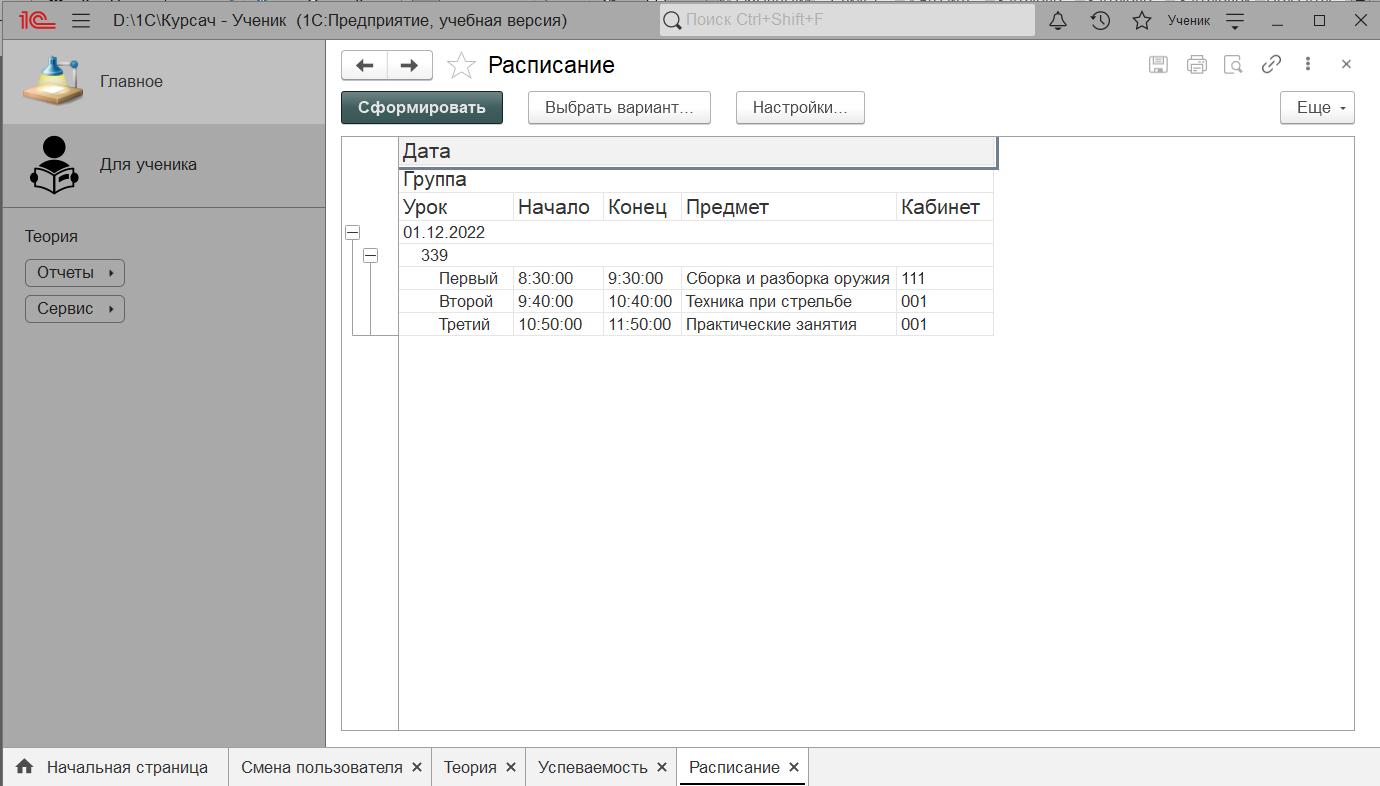


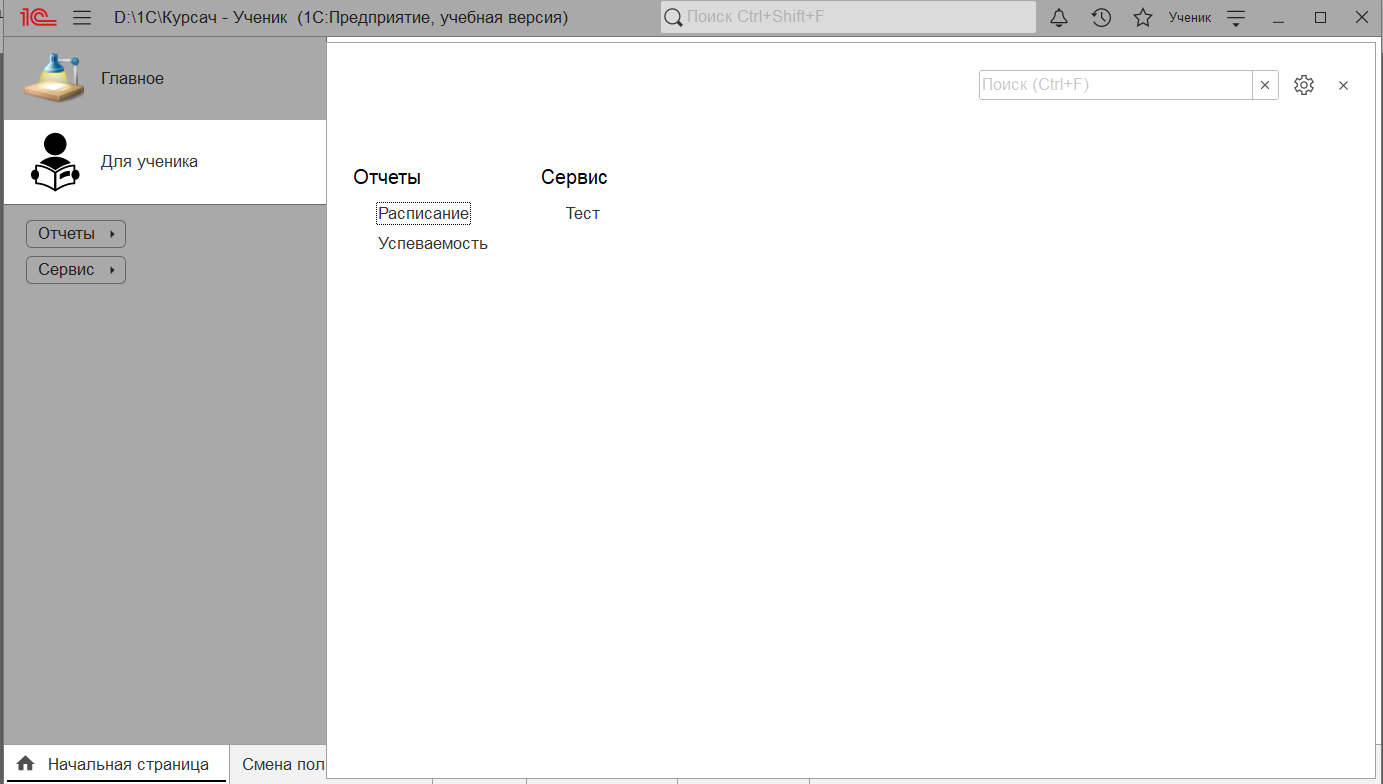


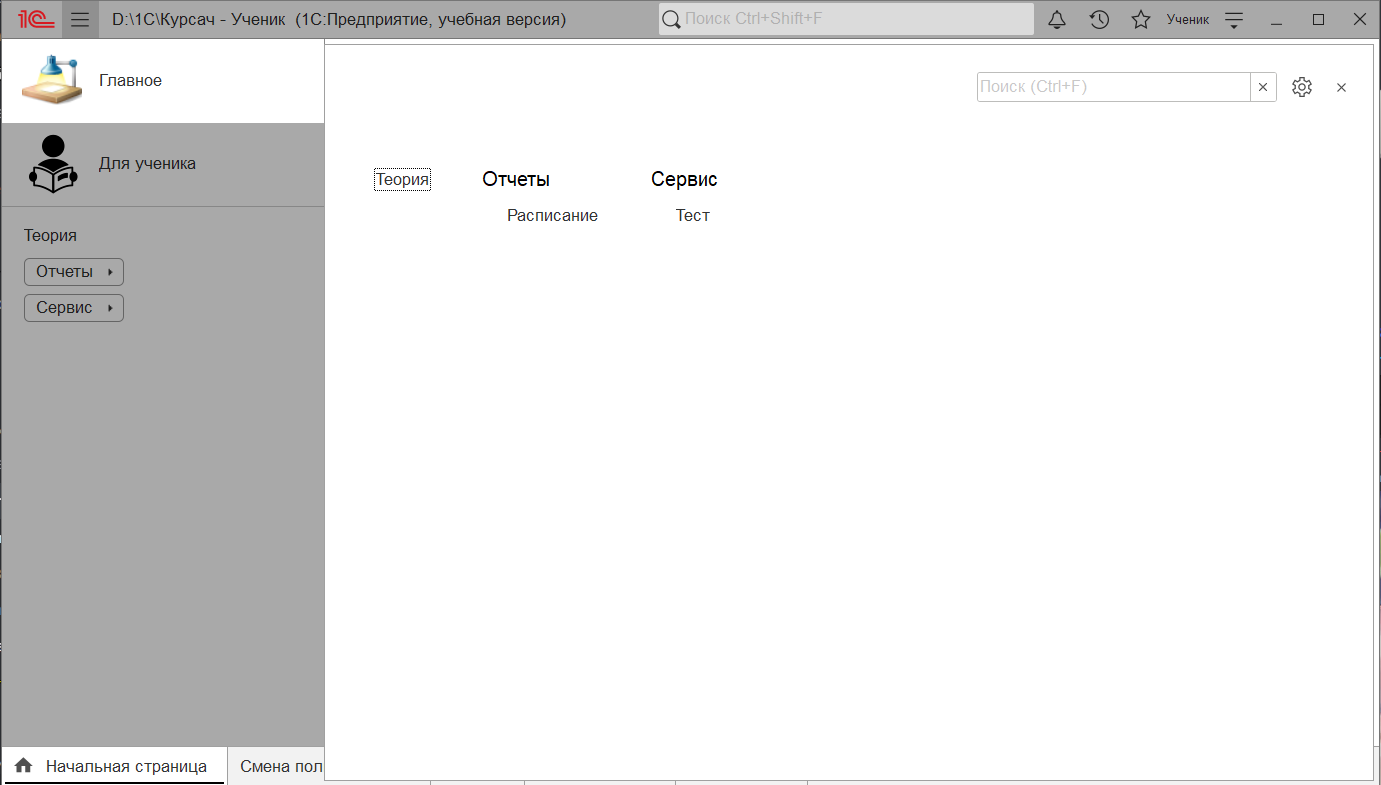


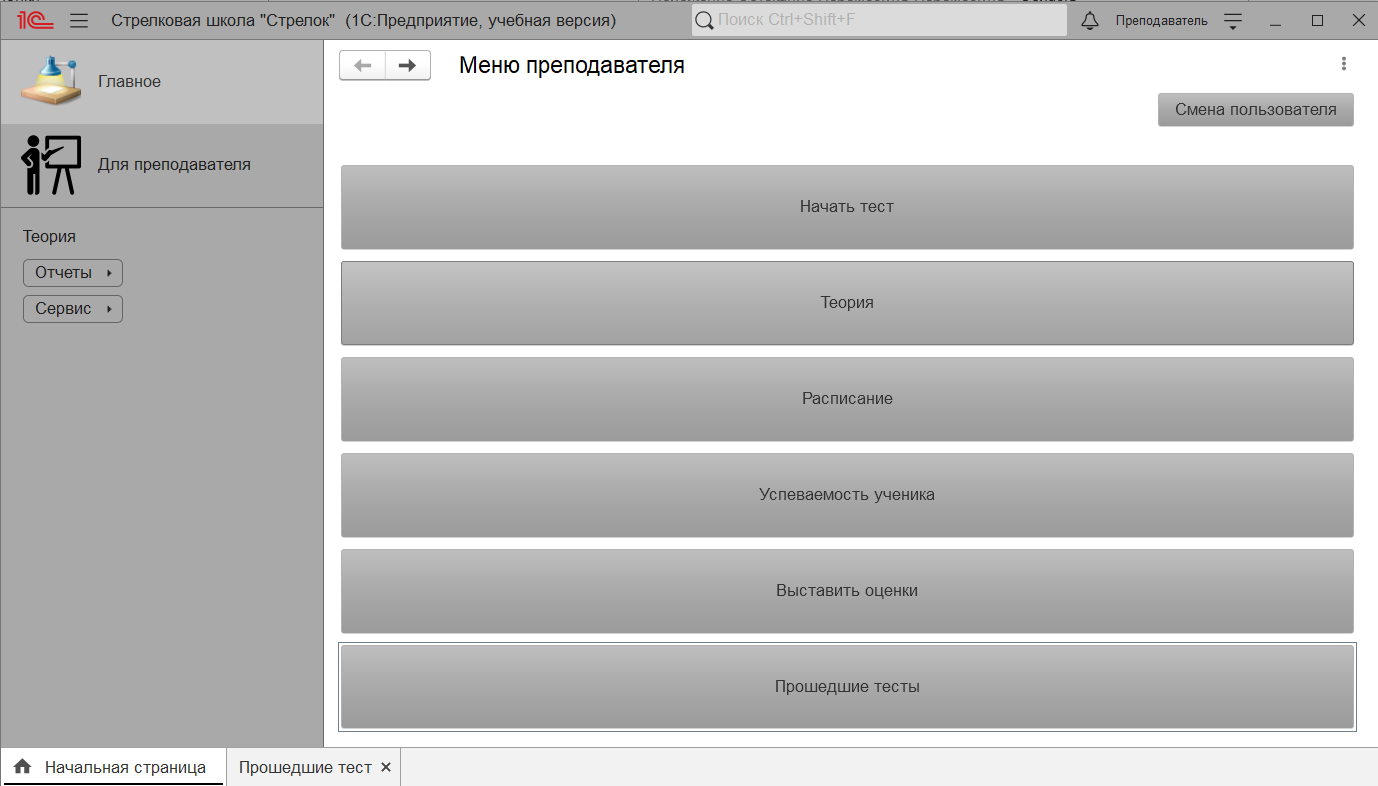


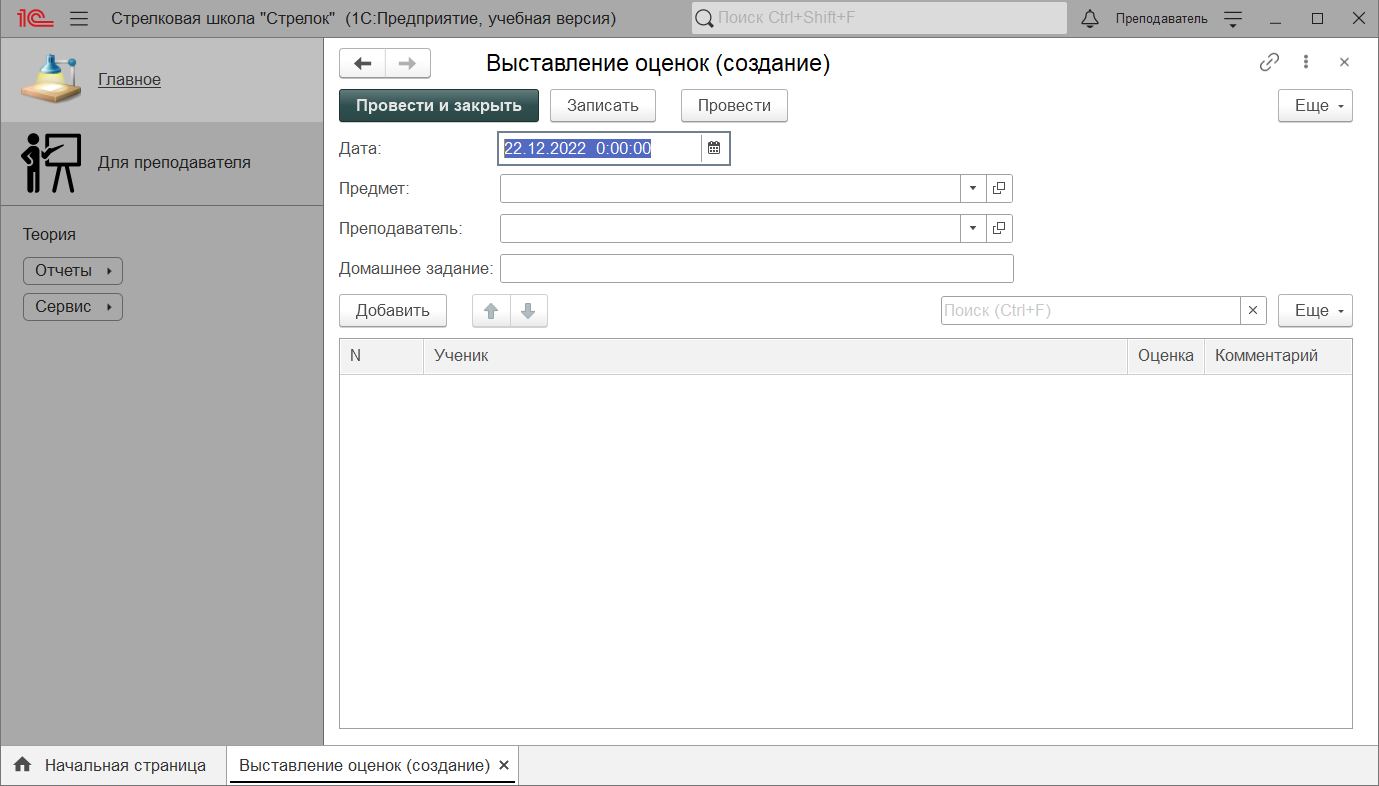


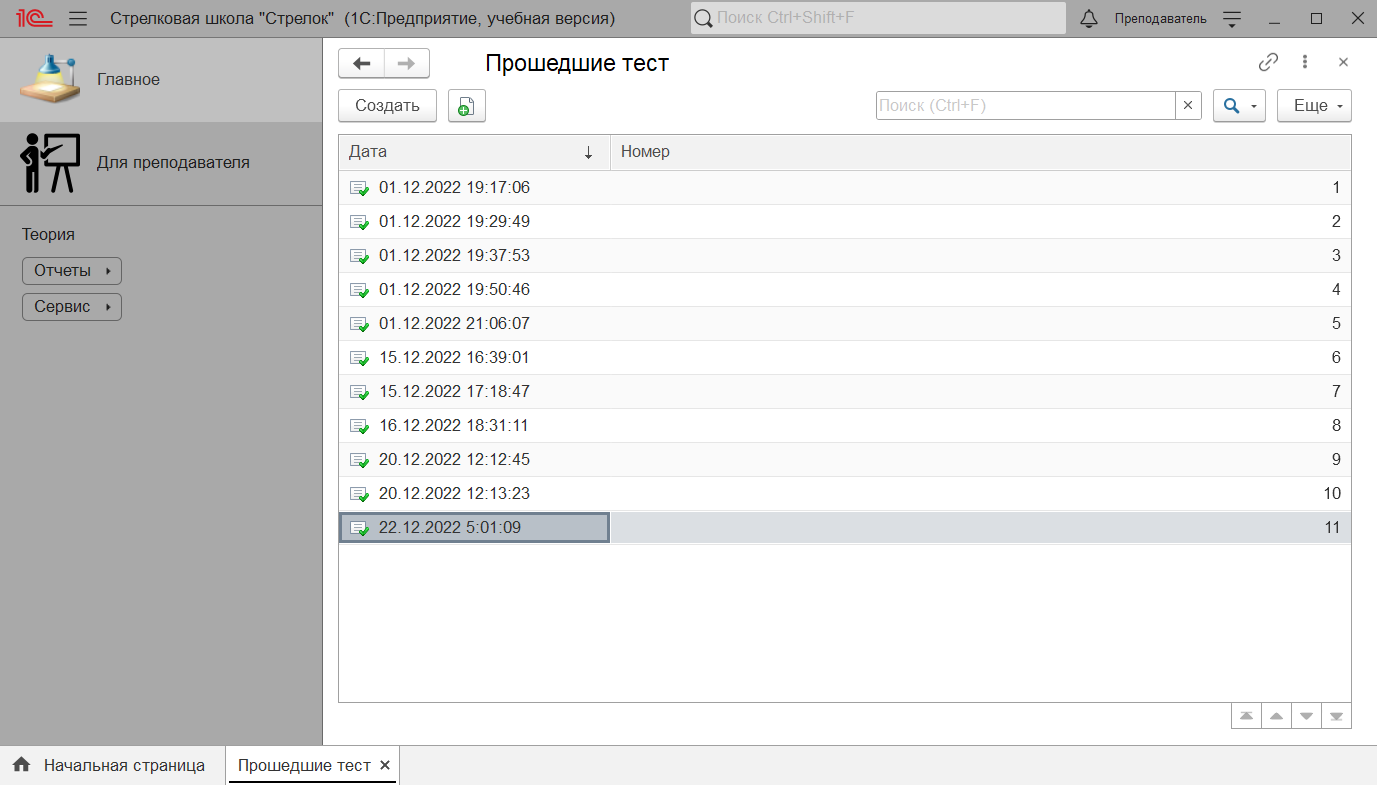


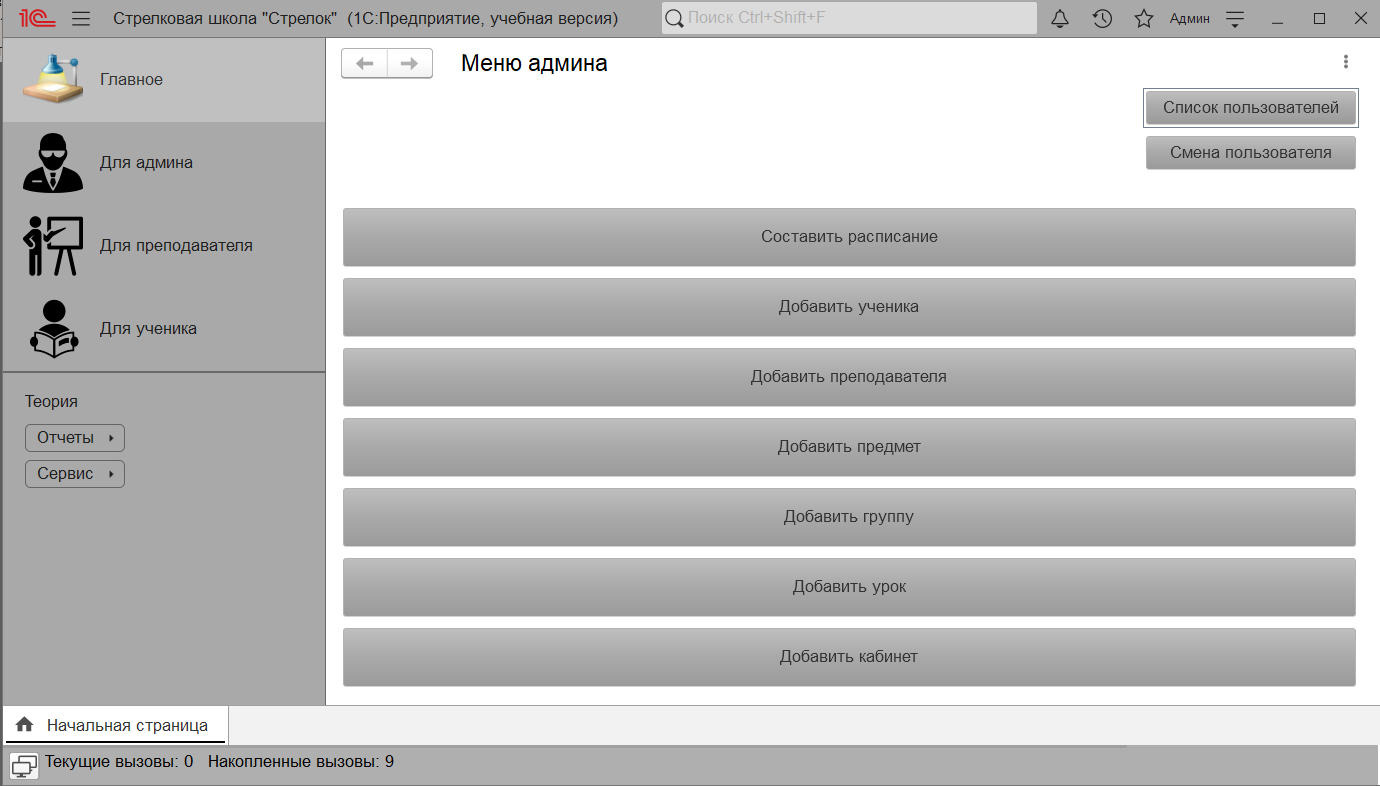


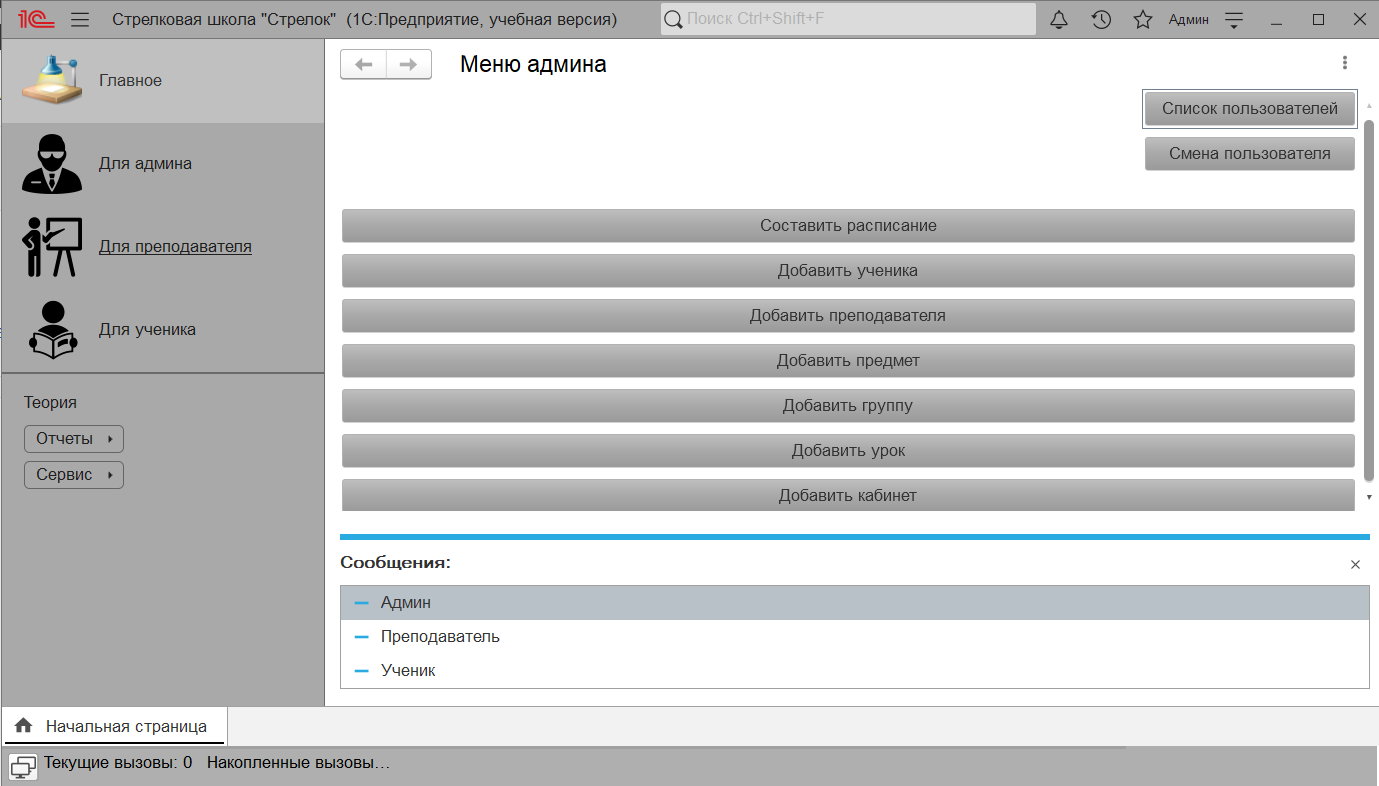


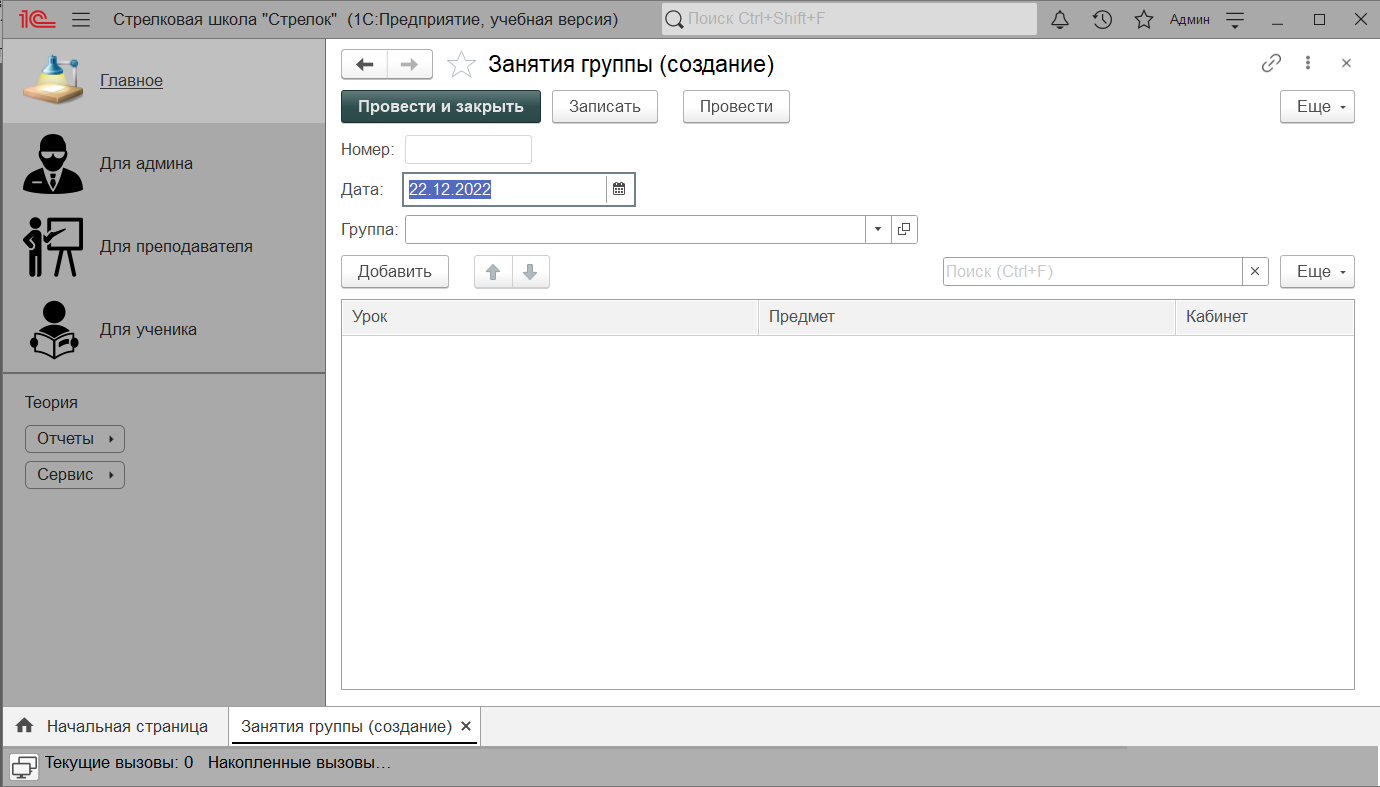


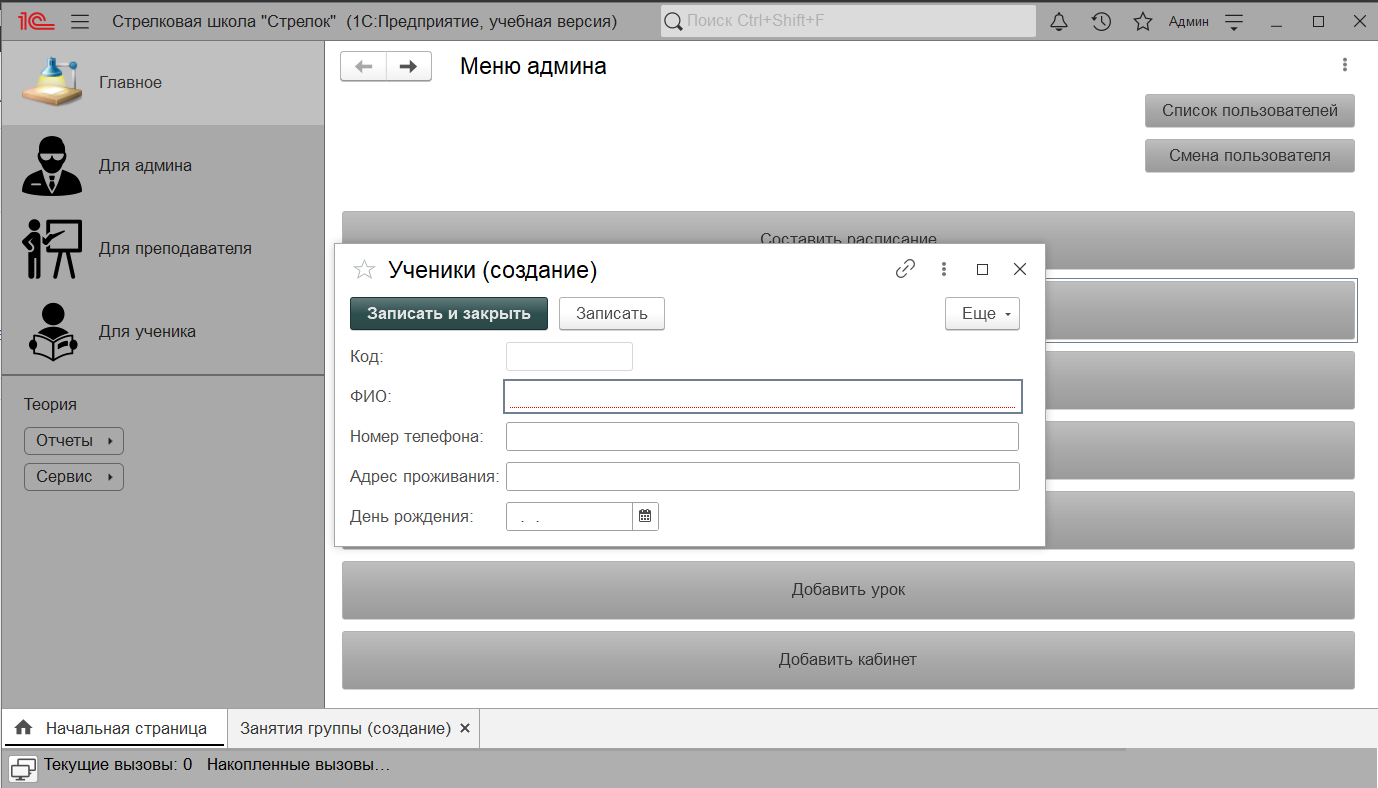


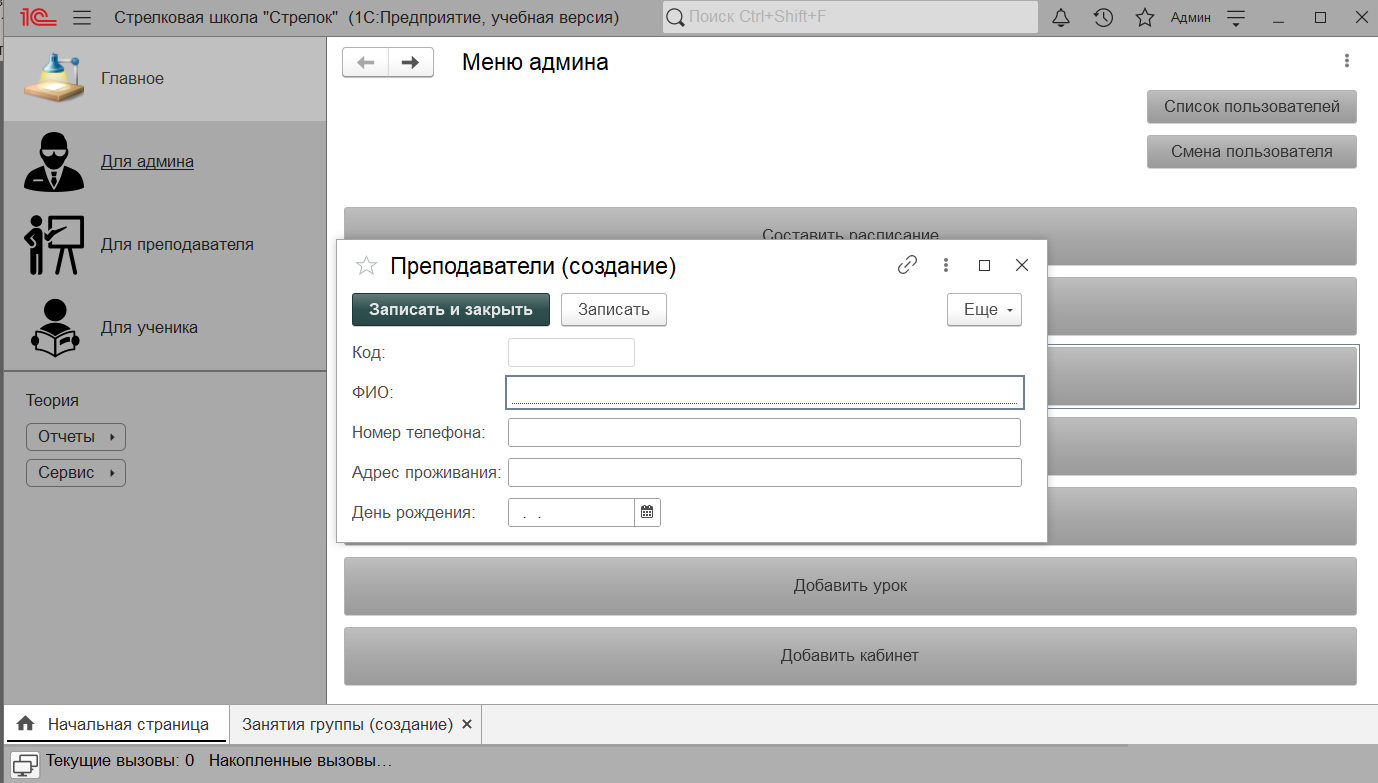


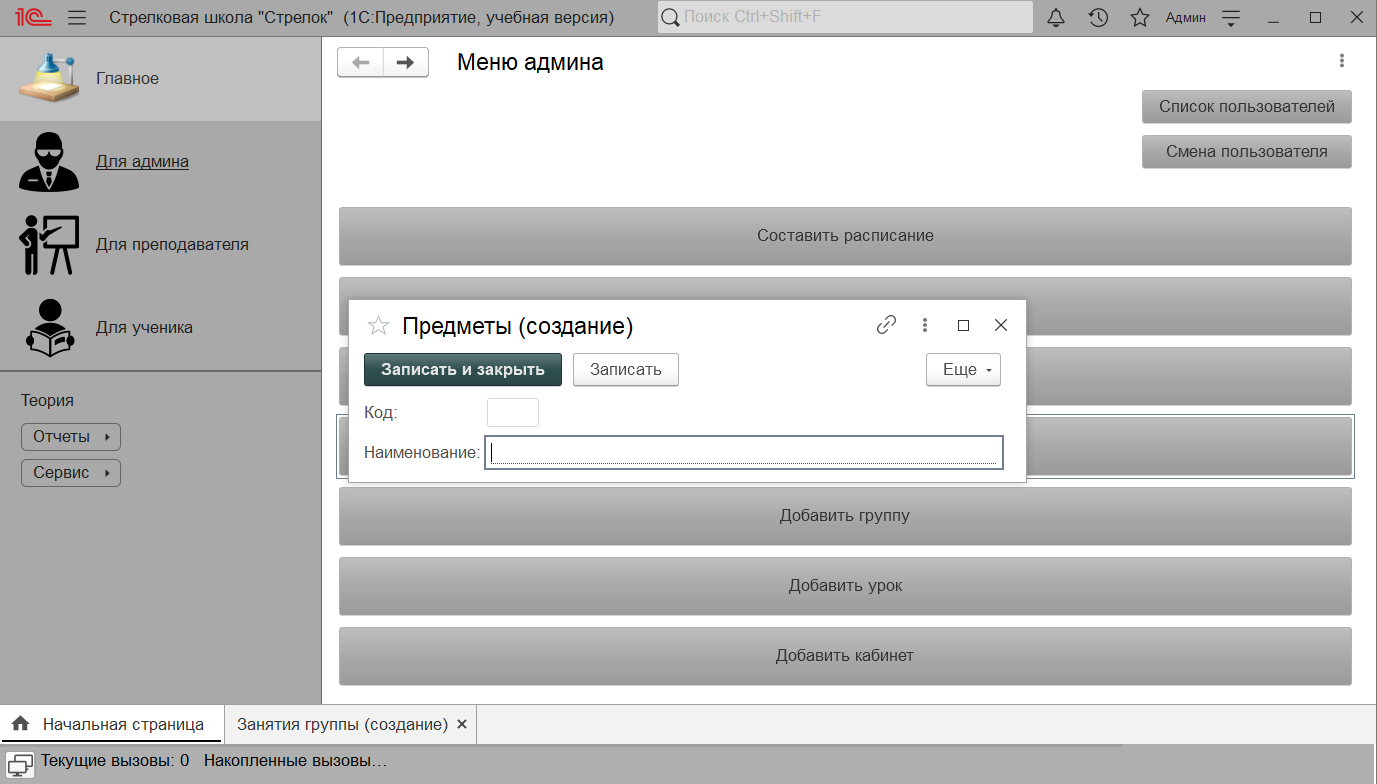


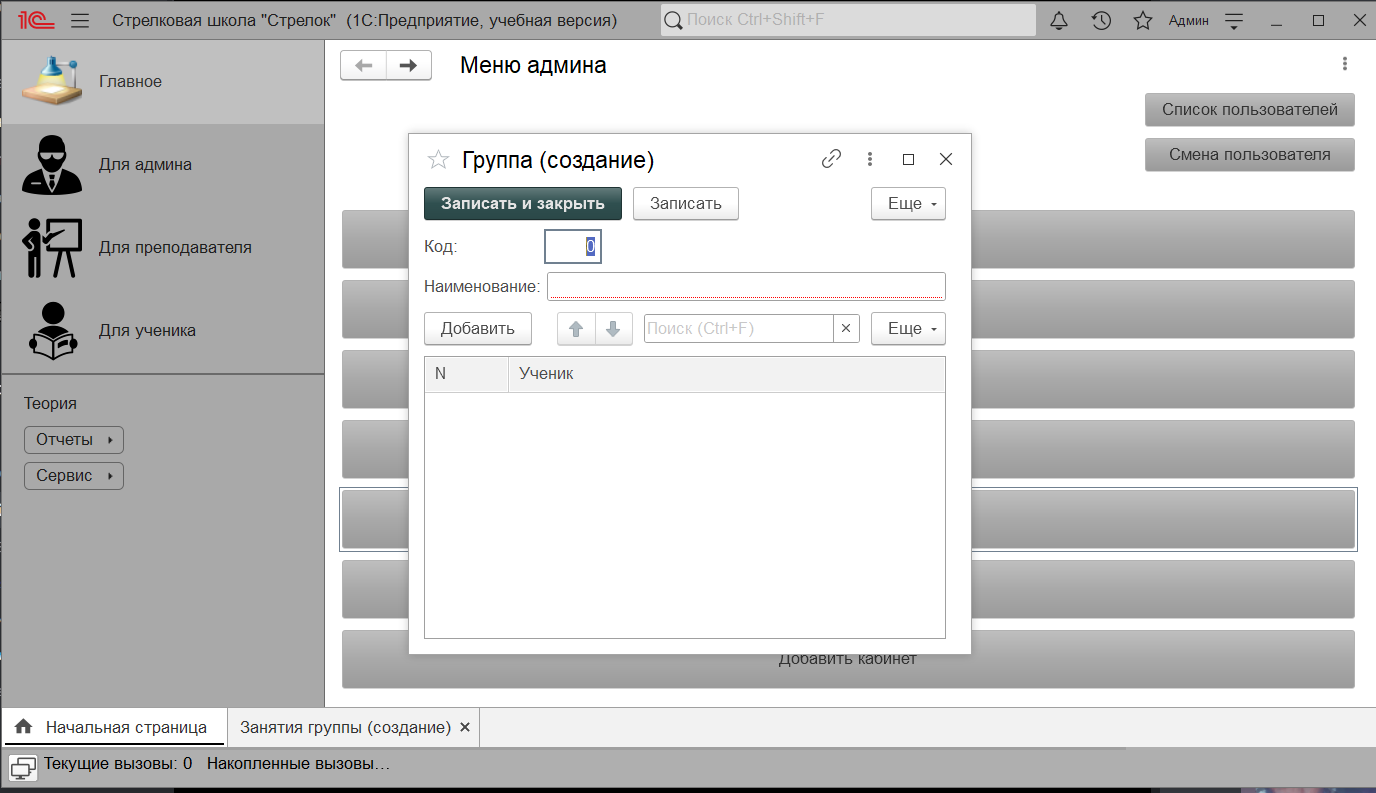


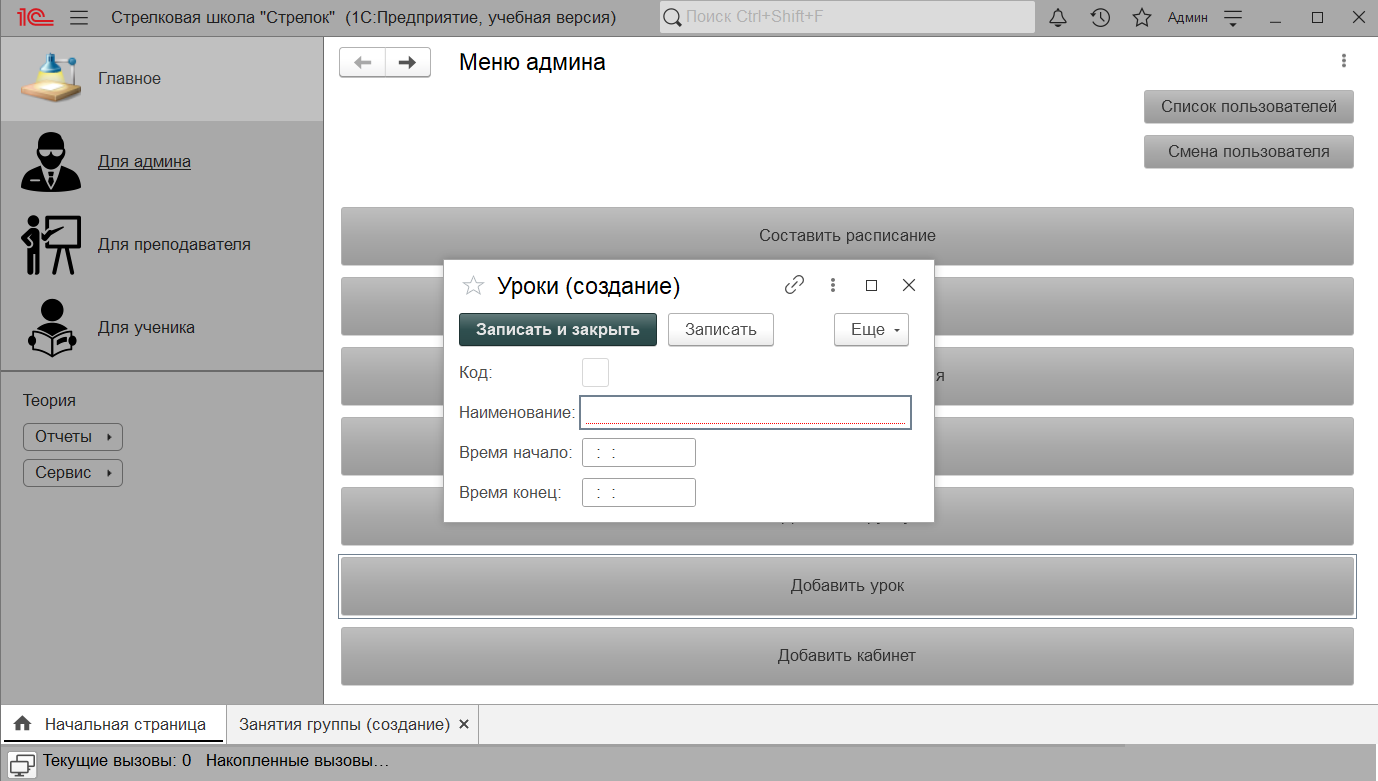


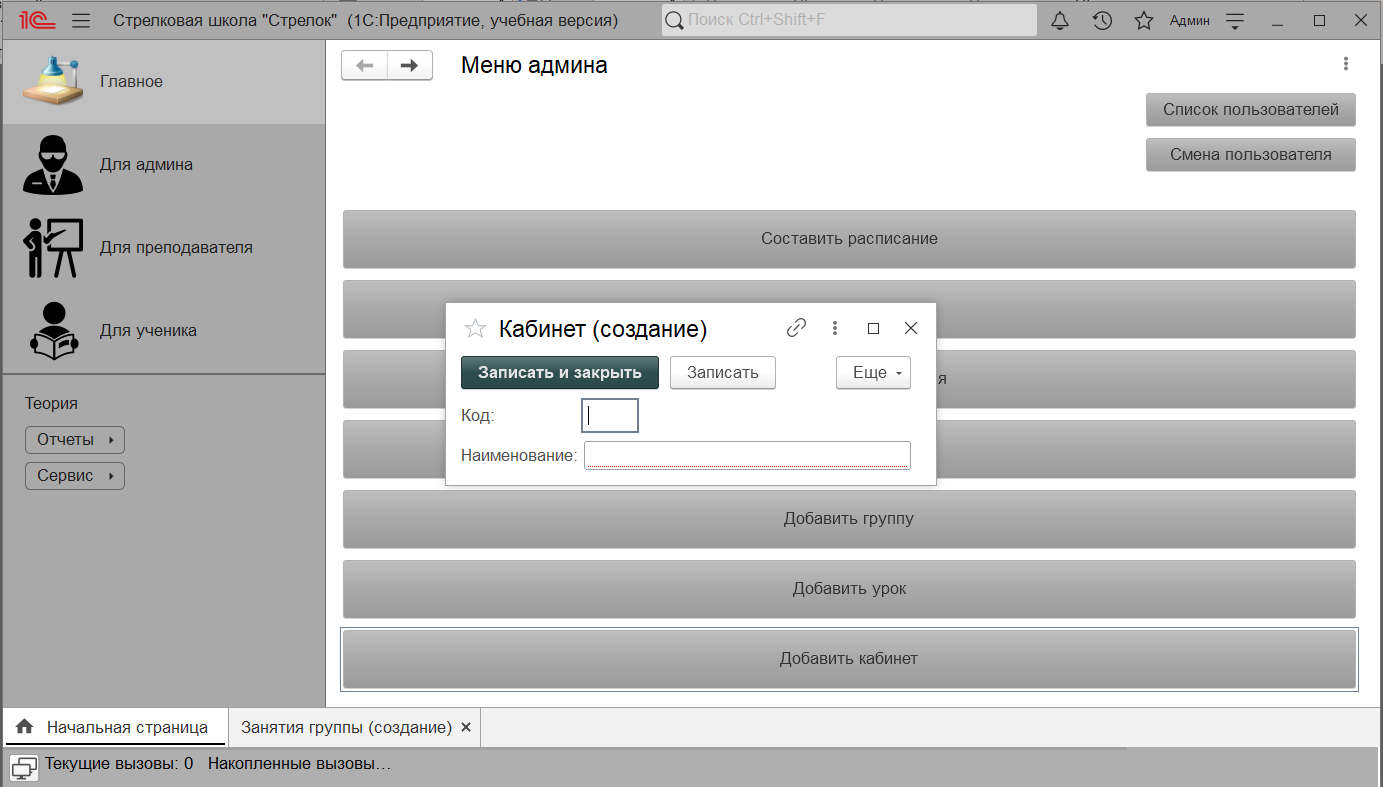


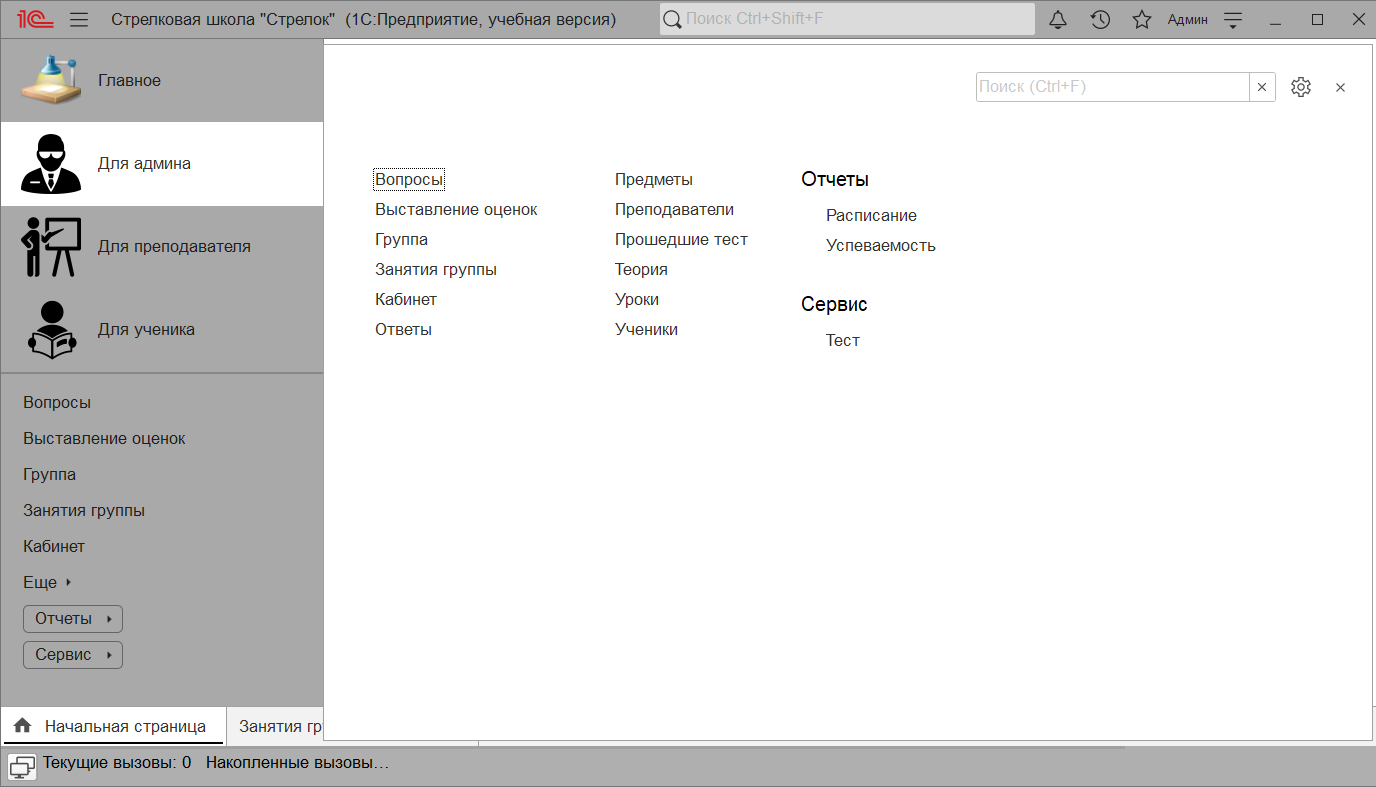












* 1. Приложение В. Листинг кода программы
     1. Модуль общей формы МенюПреподавателя:

&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;

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;

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;

ApplyVisabilityFlags();

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 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

|";

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;

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;

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, "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()

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);

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;

For Each Row In MetadataClasses Do

ClassName = Row.NameEn;

ClassNameRecent = ClassName + "Recent";

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";

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");

EndDo;

EndProcedure

Procedure ApplyLanguage()

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

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");

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");

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");

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");

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)");

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)");

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 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

&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

&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 AddFilesFromDirectory(Parent, Directory)

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

EndProcedure

&AtClient

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

If Wait = True Then

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

&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 = "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

&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

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);

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

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