

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.22.09.02.07.602.11.ПЗ

Студент

«__» _____ 20__ г.

Н.В.Коровин

Руководитель

«__» _____ 20__ г.

Д.А.Антипов

Томск 2022

СОДЕРЖАНИЕ

Введение	3
1 ОБЩАЯ ЧАСТЬ	4
1.1 Анализ предметной области	4
Приложение будет автоматизировать:	4
1.2 Выбор средств и сред разработок	6
2 СПЕЦИАЛЬНАЯ ЧАСТЬ	7
2.1 Описание требований к информационной системе	7
2.2 Диаграмма вариантов использования	9
2.3 Диаграмма состояний	10
2.4 Словарь данных	11
2.5 Пользовательские сценарии	15
2.6 Прототипы основных интерфейсов	20
3 ЗАКЛЮЧЕНИЕ	23
3.1 Приложение А. Результаты тестирования	24
3.2 Приложение Б. Результаты работы системы	30
3.3 Приложение В. Код модулей системы	44

					КП.22.09.02.07.602.11.ПЗ						
Изм	Лист	№ докум.	Подпись	Дата							
Разраб.		Коровин Н.В.			Пояснительная записка			Лит.	Лист	Листов	
Пров.		Антипов Д.А.						Т	2	111	
								ТТИТ 602 гр.			
Н. контр.											
Утв.											

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 охотничьим оружием и прохождении обучения 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.

Цель проекта: разработать информационную систему для автоматизации работ с документами в школе по обучению стрельбе;

Задачи проекта:

1. Сделать анализ предметной области
2. Выбрать средство разработки для создания системы
3. Сделать диаграммы и словарь данных
4. Сделать прототипы окон системы
5. Сделать систему с помощью выбранных средств разработки
6. Сделать пользовательские сценарии для тестов

					КП.22.09.02.07.602.11.ПЗ	Лист
						3
Изм	Лист	№ докум.	Подпись	Дата		

1 ОБЩАЯ ЧАСТЬ

1.1 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

					КП.22.09.02.07.602.11.ПЗ	Лист
						4
Изм	Лист	№ докум.	Подпись	Дата		

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 подглядывать 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 руководителями

Анализ целевой аудитории:

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

					КП.22.09.02.07.602.11.ПЗ	Лист
						5
Изм	Лист	№ докум.	Подпись	Дата		

1.2 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 C++, 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.

					КП.22.09.02.07.602.11.ПЗ	Лист
						6
Изм	Лист	№ докум.	Подпись	Дата		

2 СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1 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

Добавление картинки

1. Возможность добавления картинки в запись справочника «Теория»

					КП.22.09.02.07.602.11.ПЗ	Лист
						8
Изм	Лист	№ докум.	Подпись	Дата		

2.2 Диаграмма вариантов использования

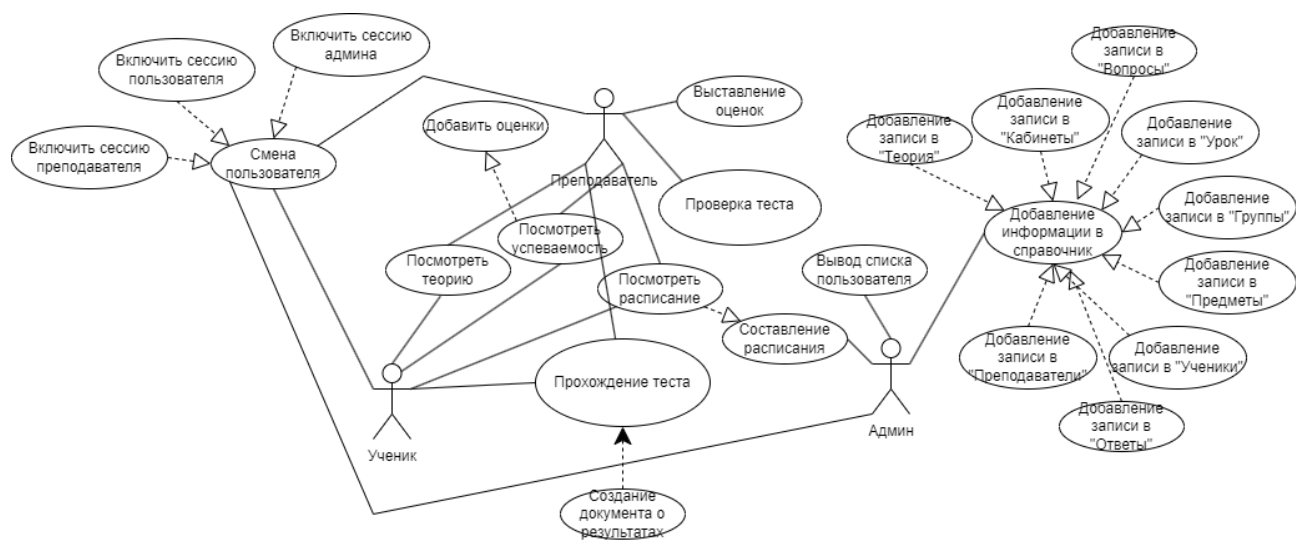


Рисунок 1. Диаграмма вариантов использования

2.3 Диаграмма состояний

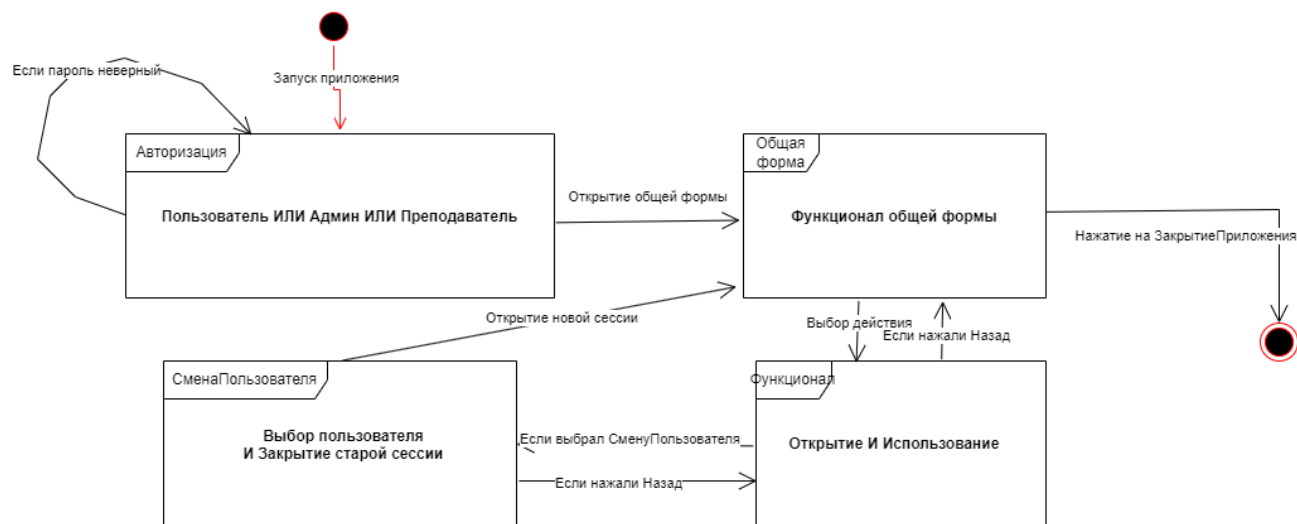


Рисунок 2. Диаграмма состояний

2.4 Словарь данных

Таблица 1 – Справочник «Вопросы»

Справочник.Вопросы	
Реквизиты	Типы данных
Код	Число(3)
Наименование	Строка(15)
ТекстВопроса	Строка(400)

Таблица 2 – Справочник «Ответы»

Справочник.Ответы	
Реквизиты	Типы данных
Код	Число(2)
Наименование	Строка(25)
ПравильныйОтвет	Булево

Таблица 3 – Справочник «Предметы»

Справочник.Предметы	
Реквизиты	Типы данных
Код	Строка(3)
Наименование	Строка(50)

Таблица 4 – Справочник «Группа»

Справочник.Группа	
Реквизиты	Типы данных
Код	Число(3)
Наименование	Строка(25)
ТабЧасть.Перечень.Ученик	СправочникСсылка.Ученики

Таблица 5 – Справочник «Ученик»

Справочник.Ученики	
Реквизиты	Типы данных
Код	Строка(9)
Наименование	Строка(100)
НомерТелефона	Строка(20)
АдресПроживания	Строка(100)
ДеньРождения	Дата(Дата)

Таблица 6 – Справочник «Уроки»

Справочник.Уроки	
Реквизиты	Типы данных
Код	Строка(1)
Наименование	Строка(25)
ВремяНачало	Дата(Время)
ВремяКонец	Дата(Время)

Таблица 7 – Справочник «Преподаватели»

Справочник.Преподаватели	
Реквизиты	Типы данных
Код	Строка(9)
Наименование	Строка(100)
НомерТелефона	Строка(20)
АдресПроживания	Строка(100)
ДеньРождения	Дата(Дата)

Таблица 8 – Справочник «Теория»

Справочник.Теория	
Реквизиты	Типы данных
Код	Число(9)
Наименование	Строка(25)
Текст	Строка(Неогранич.длина)
Картинка	ХранилищеЗначения

Таблица 9 – Справочник «Кабинет»

Справочник.Кабинет	
Реквизиты	Типы данных
Код	Строка(3)
Наименование	Строка(25)

Таблица 10 – Документ «ПрошедшиеТест»

Документ.ПрошедшиеТест	
Реквизиты	Типы данных
ФИО	СправочникСсылка.Ученики
КоличествоБаллов	Число(10)

Таблица 11 – Документ «ЗанятияГруппы»

Документ.ЗанятияГруппы	
Реквизиты	Типы данных
Группа	СправочникСсылка.Группа
ТабЧасть.Перечень.Урок	СправочникСсылка.Уроки
ТабЧасть.Перечень.Предмет	СправочникСсылка.Предметы
ТабЧасть.Перечень.Кабинет	СправочникСсылка.Кабинет

Таблица 12 – Документ «ВыставлениеОценок»

Документ.ВыставлениеОценок	
Реквизиты	Типы данных
Предмет	Строка(9)
Преподаватель	Строка(100)
ДомашнееЗадание	Строка(20)
ТабЧасть.Перечень.Ученик	Строка(100)
ТабЧасть.Перечень.Оценка	Число(1)
ТабЧасть.Перечень.Комментарий	Строка(10)

2.5 Пользовательские сценарии

Проверять систему будем 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, которая разрабатывается для курсовой работы, будет методология “Белый ящик”, так как это методология подразумевает открытый код для разработчика.

Таблица 13 - Test case №1

Test Case №	ТС_1
Приоритет теста	Высокий
Название тестирования/Имя	Логин: 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»
Тестовые данные	Ученик, преподаватель, админ
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.

Таблица 14 - 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	<ol style="list-style-type: none"> 1. Запуск программы под пользователем «Преподаватель». 2. Нажимаем кнопку «Теория», после того, как выбрали запись в справочнике, нажимаем на поле картинки, чтобы попробовать загрузить. 3. Загружаем 2 разные расширения файла фотографии.
Тестовые данные	<p>Рисунок 3 - Картинка.jpg</p>  <p>Рисунок 4 - Картинка.png</p> 
Ожидаемый результат	Две этих фотографий будут подходить для загрузки в справочник.
Фактический результат	В окне выбора видно лишь фотографию с расширением .jpg
Предпосылки	Добавило лишь одну фотографию, так как можно только .jpg формат, так как в коде это упущено.
Статус (Зачет/Незачет)	Незачет
Комментарии	<ol style="list-style-type: none"> 1. Для тестирования были взяты 2 фотографии, одна в .jpg расширении, другая в .png, но загрузить получилось в .jpg

Таблица 15 - 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.
Тестовые данные	Имя пользователя: Преподаватель, ученик, админ
Шаги тестирования	2. Запуск программы под любым пользователем. 3. Нажимаем кнопку «Смена пользователя», после перехода на другую страницу в поле «Имя пользователя» выбираем пользователя. 4. 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	

Таблица 16 - Test case №4

Test Case №	ТС_4
Приоритет теста	Среднее
Название тестирования/Имя	Проверка функции «Тест»
Резюме испытания	Пользователь может пройти тест.
Шаги тестирования	<ol style="list-style-type: none"> 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». 4. Нажимаем варианты ответы и нажимаем «Вперед» пока не не появится конца «Завершить тест» 5. Открывается окно и нажимаем на кнопку «Да», чтобы подтвердить завершение теста.
Тестовые данные	ФИО ученика: Кунтиков Павел Михайлович, Выбранные варианты ответа: «прицел», «на спусковом крючке».
Ожидаемый результат	Появится результат с количеством правильных ответов и покажется сообщение о создании документа о прохождении теста данным учеником.
Фактический результат	Появился результат и сообщение про документ.
Предпосылки	Чтобы проверить создаться документ необходимо зайти в документ «Прошедшие тест».
Статус (Зачет/Незачет)	Зачет
Комментарии	Так как тестируем по методологии «Белого ящика», то при этих вариантах должно выйти сообщение про все правильные ответы, которые выбрали.

Таблица 17 - Test case №5

Test Case №	ТС_5
Приоритет теста	Среднее
Название тестирования/Имя	Проверка добавления оценок и их отображение
Резюме испытания	Пользователь может добавить оценку и их посмотреть в отчете.
Шаги тестирования	<ol style="list-style-type: none"> 1. Запуск программы под пользователем «Преподаватель» 2. Нажимаем кнопку «Выставление оценок» 3. Выбираем «предмет», «преподавателя», текст домашнего задания и ученика, которому будем ставить оценку. 4. После добавления в поля данных нажимаем «Провести и закрыть» 5. Нажимаем на кнопку успеваемость, выбираем «ученика», которому поставили оценку, и проверяем наличие поставленной оценки
Тестовые данные	<p>Предмет: Практические занятия</p> <p>Преподаватель: Коршунов Алексей Михайлович</p> <p>Домашнее задание: ///</p> <p>Ученик: Романенко Алексей Романович</p> <p>Оценки: 9; 5; 0; 1</p>
Ожидаемый результат	Не должно позволить добавление оценки не равняющему условию Оценка > 0 И (Оценка < 5 ИЛИ Оценка = 5)
Фактический результат	Записало все оценки, которые были взяты записали для теста
Предпосылки	В добавлении оценки отсутствует это условие, поэтому оно и добавило эти оценки.
Статус (Зачет/Незачет)	Не зачет
Комментарии	

2.6 Прототипы основных интерфейсов

Прототип окна админа «Стрелковая школа «Стрелок» - Админ». Интерфейс имеет стандартное оформление окна с заголовком, кнопками управления (минимизировать, maximize, закрыть) и панелью быстрого запуска.

Заголовок: Стрелковая школа «Стрелок» - Админ

Панель быстрого запуска:

- Изображение: ☐ Для админа
- Время: 18:00:00
- Список пользователей
- Смена пользователя

Справочники:

- Ученики
- Преподаватели
- Предметы
- Уроки
- Группы
- Кабинеты
- Еще

Кнопки действий:

- Начать тест
- Добавить ученика
- Добавить преподавателя
- Добавить предмет
- Добавить урок
- Добавить группу
- Добавить кабинет

Рисунок 5 - Прототип окна админа

Стрелковая школа "Стрелок" - Преподаватель

Изображение

Для преподавателя

Время: 18:00:00

Смена пользователя

Начать тест

Теория

Успеваемость ученика

Расписание

Выставить оценки

Прошедшие тесты

Справочники:

Ученики

Преподаватели

Предметы

Уроки

Группы

Кабинеты

Еще

Рисунок 6 - Прототип окна преподавателя

Стрелковая школа "Стрелок" - Ученик

Изображение

Для ученика

Справочники:

Ученики

Преподаватели

Предметы

Уроки

Группы

Кабинеты

Еще

Время: 18:00:00

Смена пользователя

Начать тест

Теория

Успеваемость

Расписание

Рисунок 7 - Прототип окна ученика

3 ЗАКЛЮЧЕНИЕ

Эта система выполняет свою цель, которая была поставлена перед разработкой, а именно помогает в эффективности введения работы с отчетами и документами. Упростит просмотр информации о занятиях, успеваемости и результатах прохождений теста. Хотя система сделана не идеальна, но были сделаны небольшие недочеты, но и с этими недочетами можно работать с ней. Систему в дальнейшем можно привести к идеальному её состоянию.

					КП.22.09.02.07.602.11.ПЗ	Лист
						23
Изм	Лист	№ докум.	Подпись	Дата		

3.1 Результаты тестирования

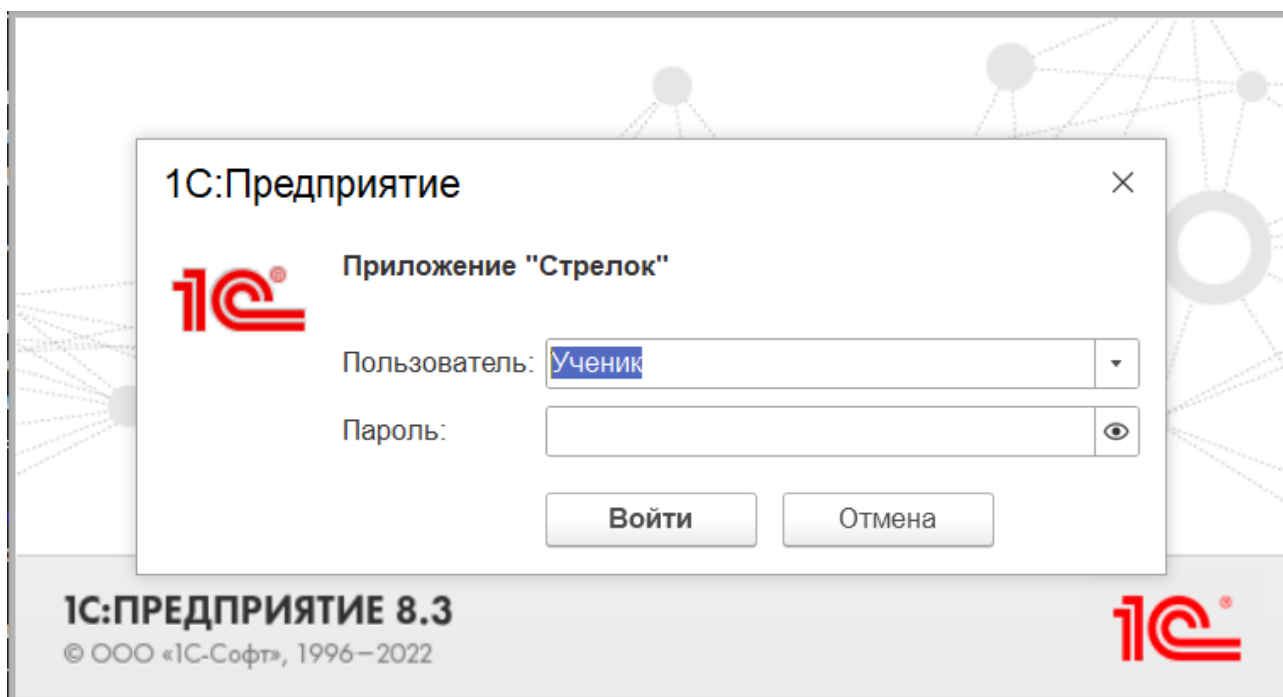


Рисунок 8 Тест №1

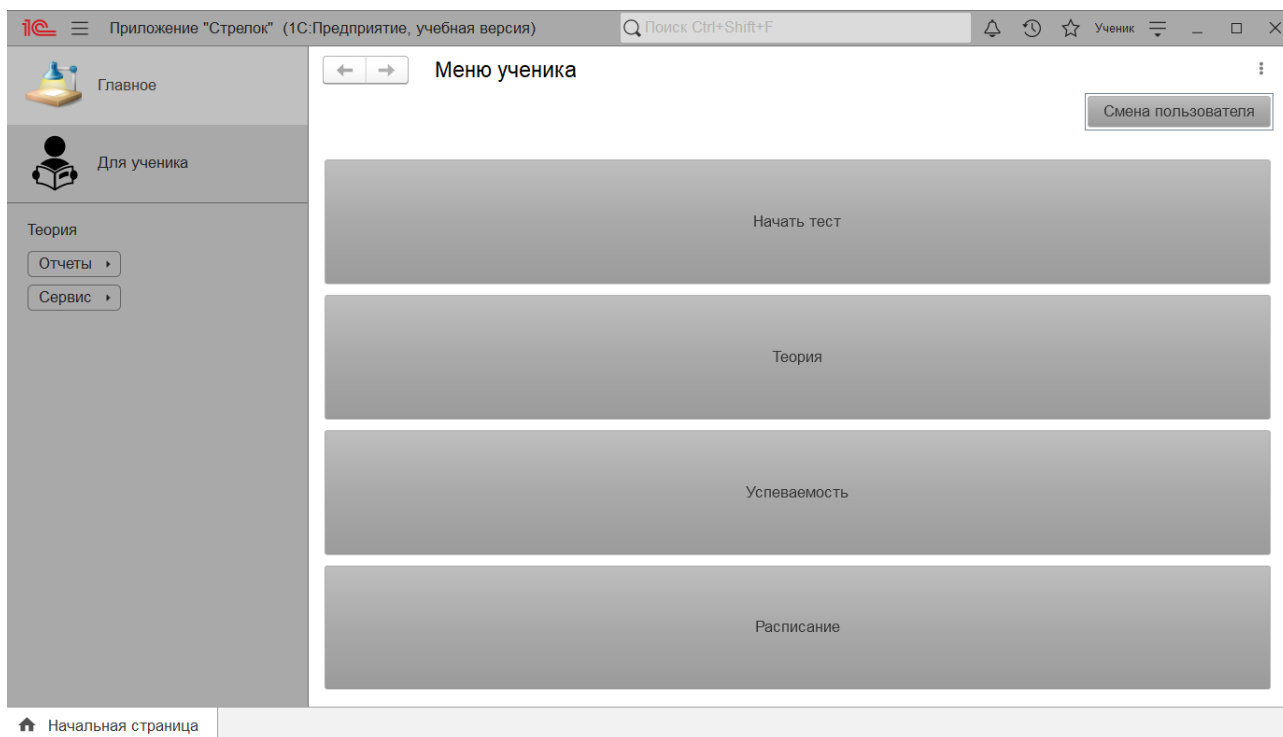


Рисунок 9 Тест №1

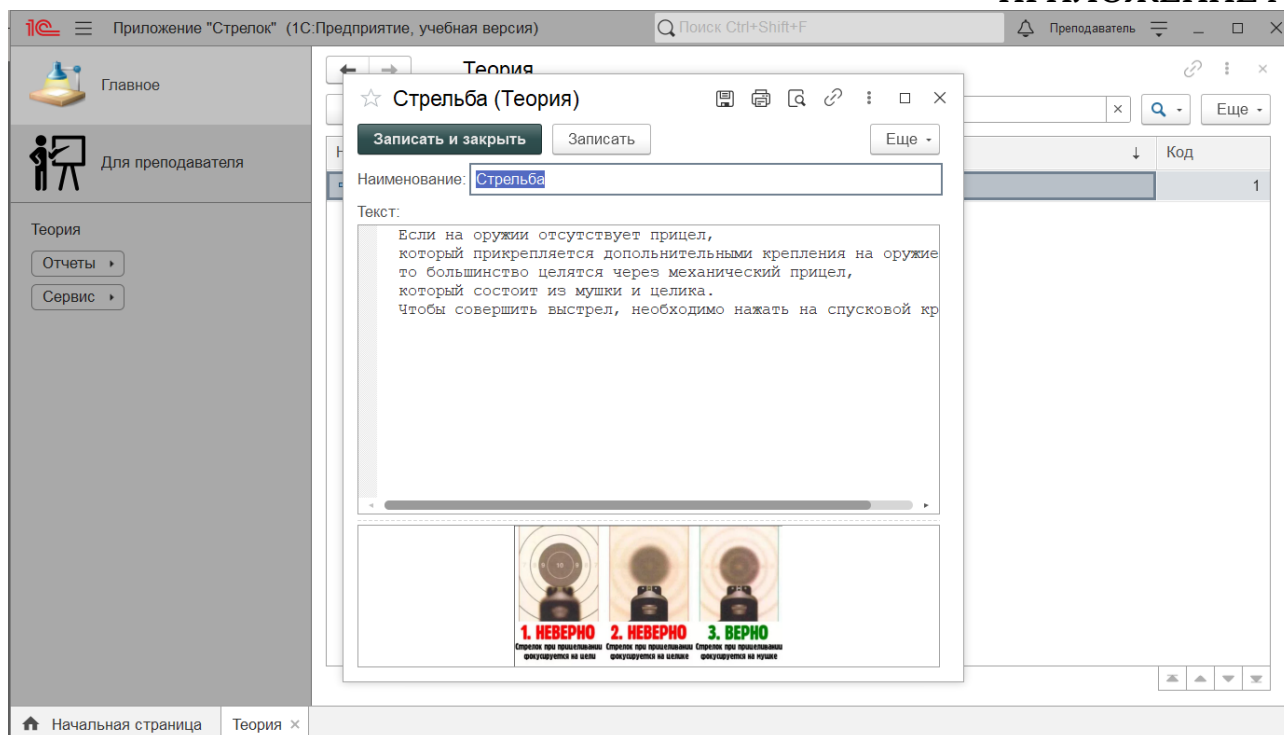


Рисунок 10 Тест №2

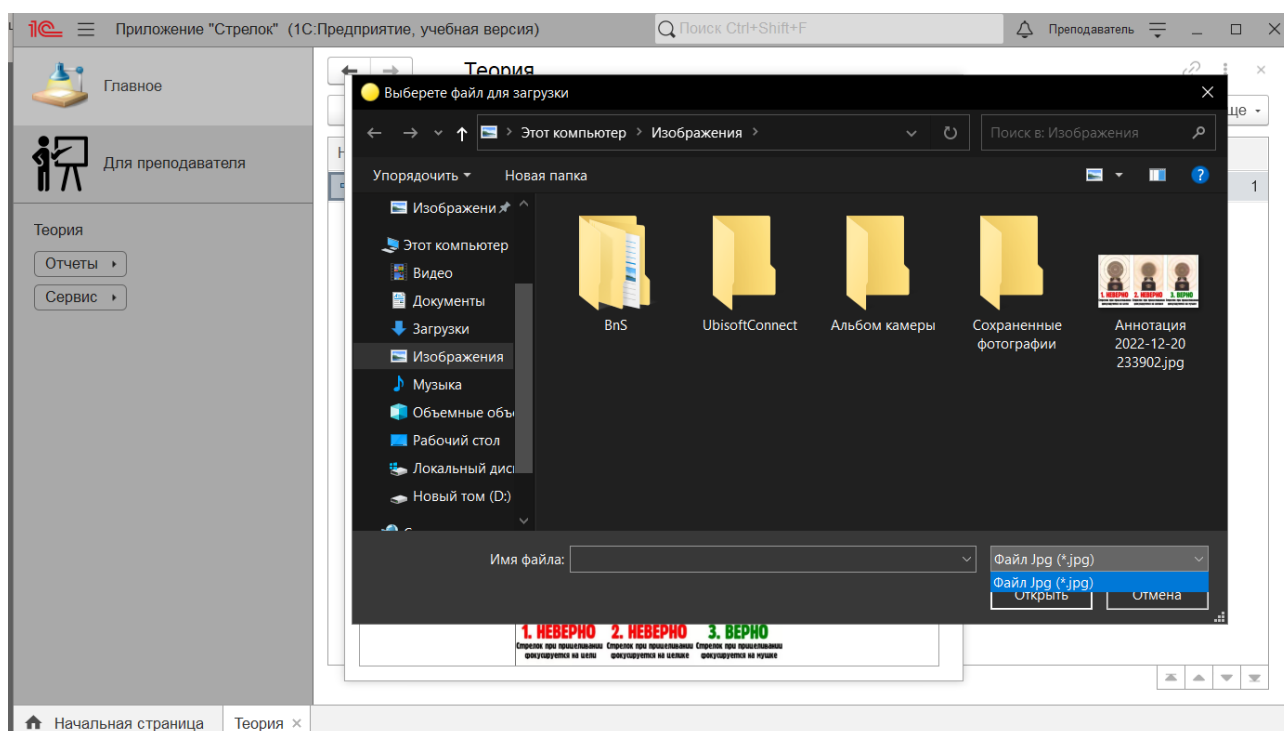


Рисунок 11 Тест №2

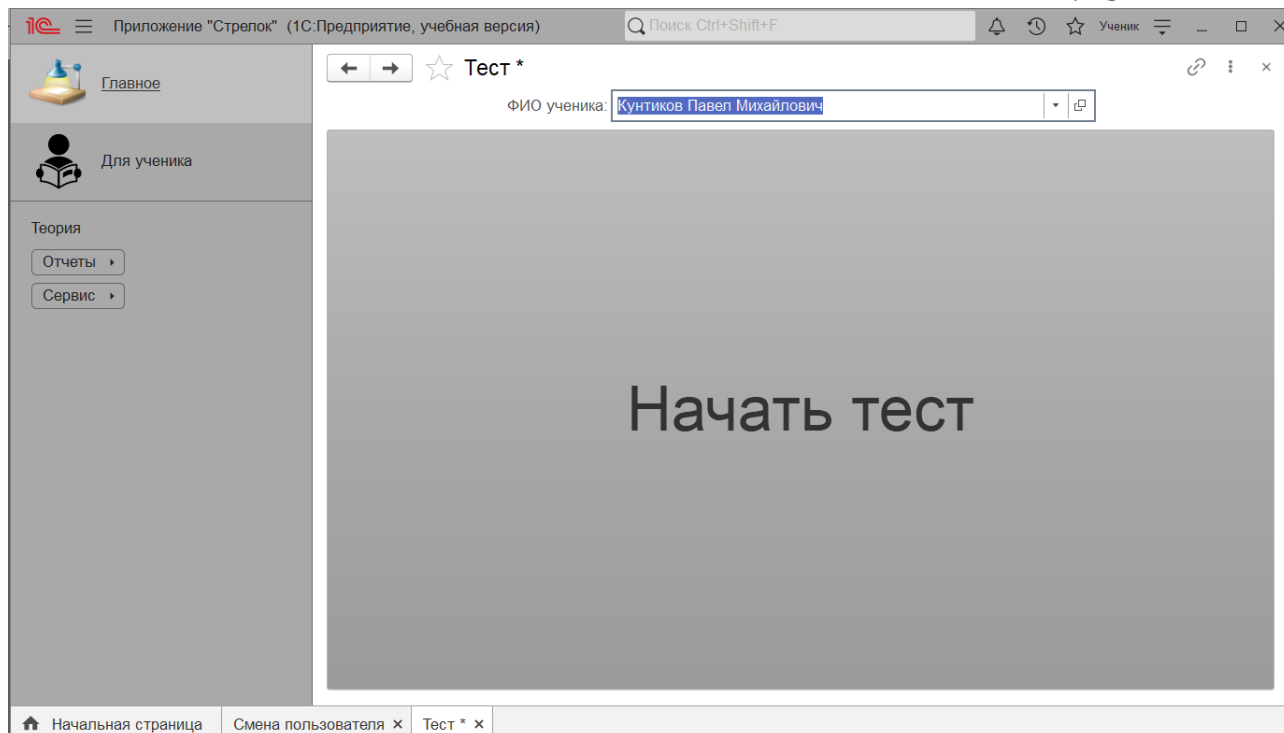


Рисунок 12 Тест №3

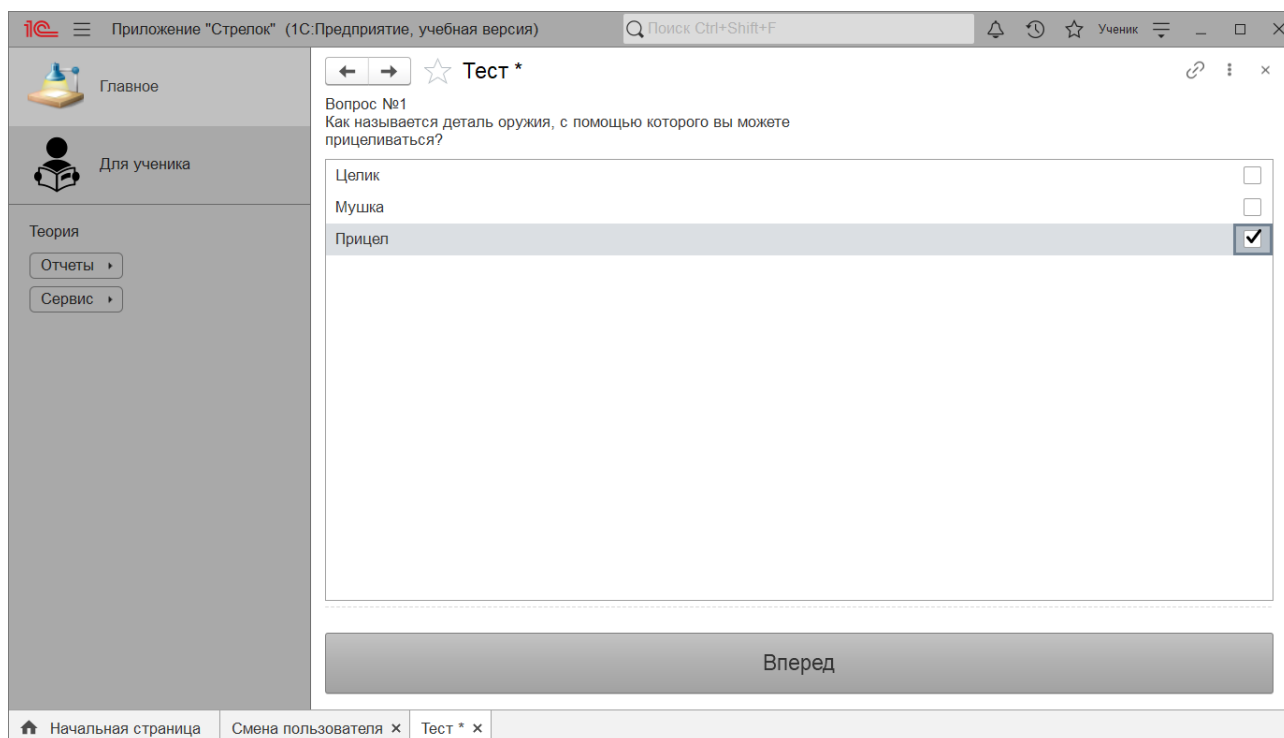


Рисунок 13 Тест №3

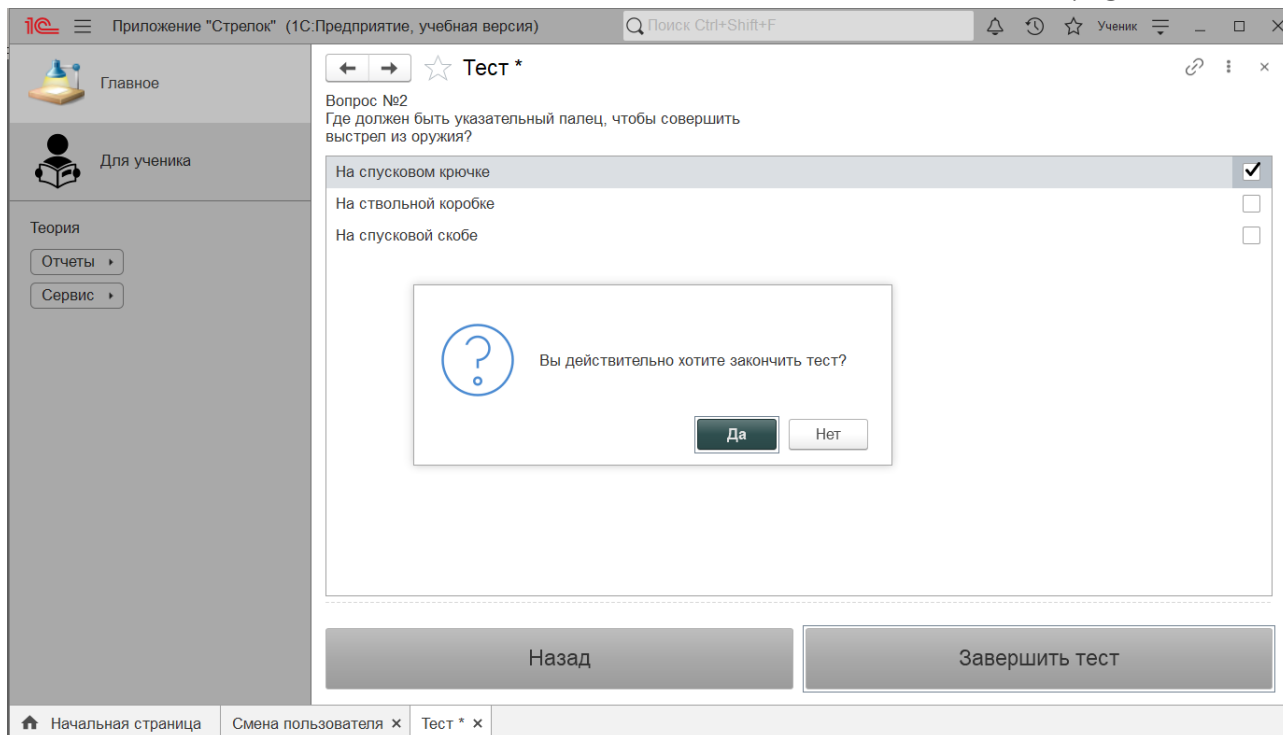


Рисунок 14 Тест №3

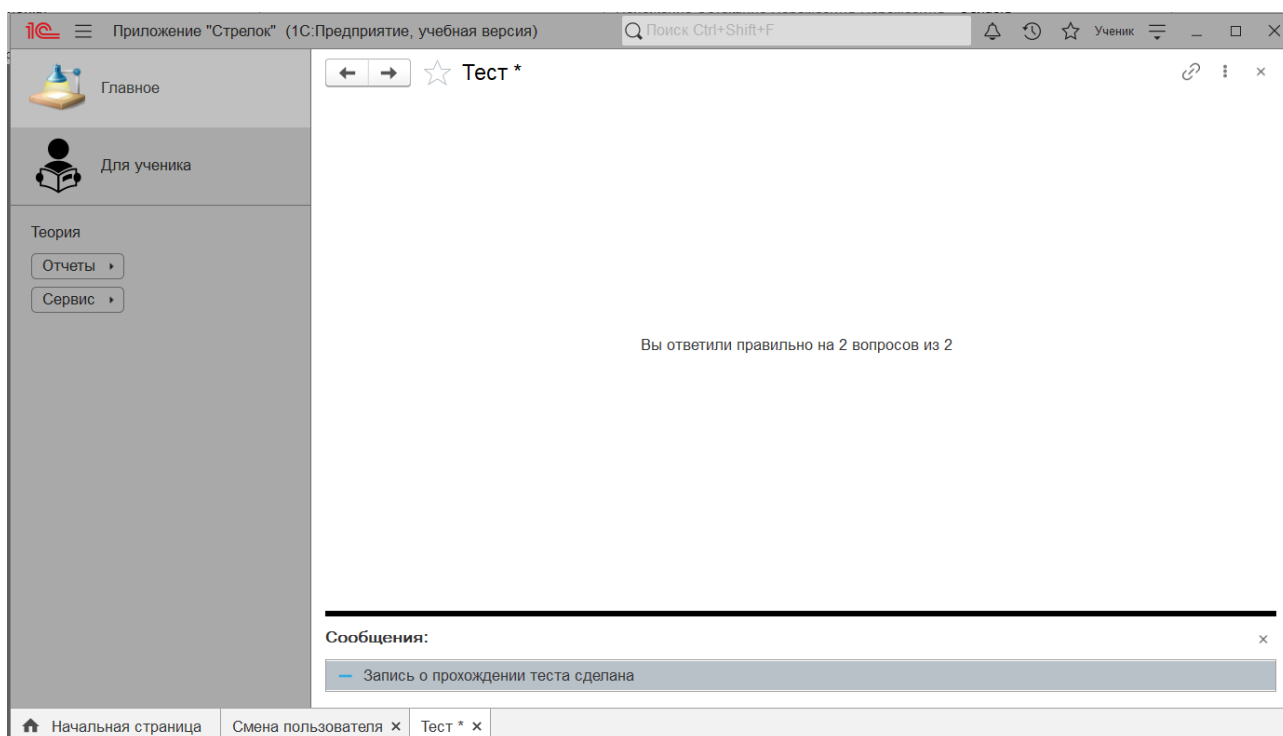


Рисунок 15 Тест №3

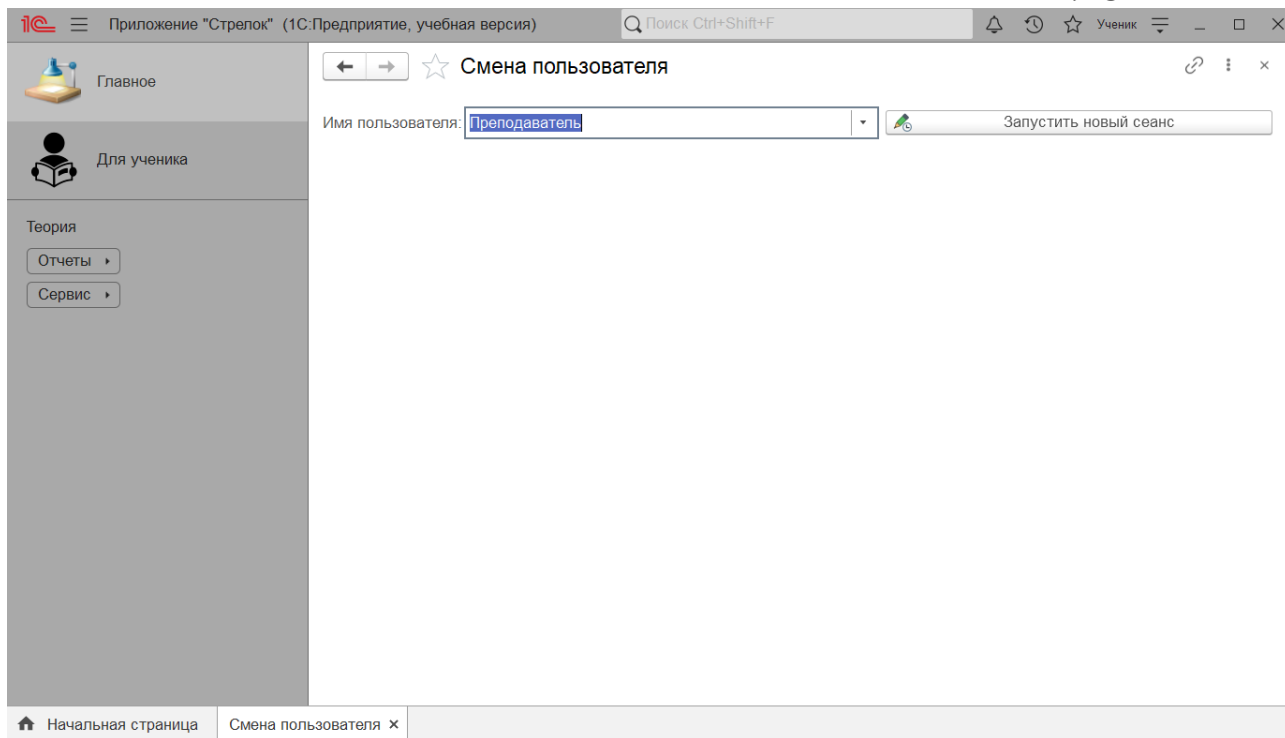


Рисунок 16 Тест №4

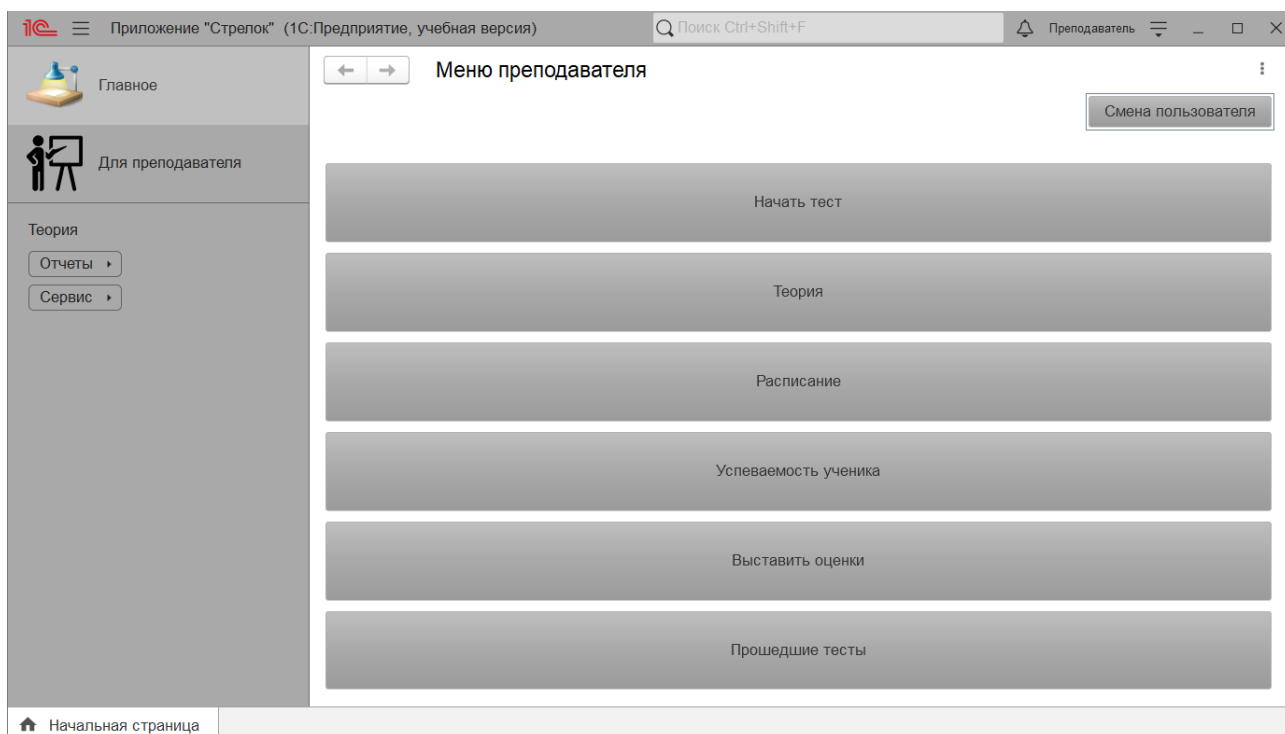


Рисунок 17 Тест №4

Приложение "Стрелок" (1С:Предприятие, учебная версия) Поиск Ctrl+Shift+F Преподаватель

Главное Для преподавателя Теория Отчеты Сервис

Выставление оценок (создание) *

Провести и закрыть Записать Провести Еще -

Дата: 24.12.2022 0:00:00

Предмет: Практические занятия

Преподаватель: Коршунов Алексей Михайлович

Домашнее задание: ///

Добавить Поиск (Ctrl+F) Еще -

N	Ученик	Оценка	Комментарий
1	Романенко Алексей Романович	9	
2	Романенко Алексей Романович	5	
3	Романенко Алексей Романович	1	
4	Романенко Алексей Романович		

Начальная страница Выставление оценок (создание) *

Рисунок 18 Тест №5

Приложение "Стрелок" (1С:Предприятие, учебная версия) Поиск Ctrl+Shift+F Преподаватель

Главное Для преподавателя Теория Отчеты Сервис

Успеваемость

Сформировать Выбрать вариант... Настройки... Еще -

Ученик: ☒ В списке Романенко Алексей Романович

Отбор: Ученик В списке "Романенко Алексей Романов..."

Предмет	Средняя оценка
Практические занятия	3,75

Дата	Предмет	Оценка	Комментарий
24.12.2022 19:47:07	Практические занятия	9	
24.12.2022 19:47:07	Практические занятия	5	
24.12.2022 19:47:07	Практические занятия		
24.12.2022 19:47:07	Практические занятия	1	

Начальная страница Успеваемость *

Рисунок 19 Тест №5

3.2 Результаты работы системы

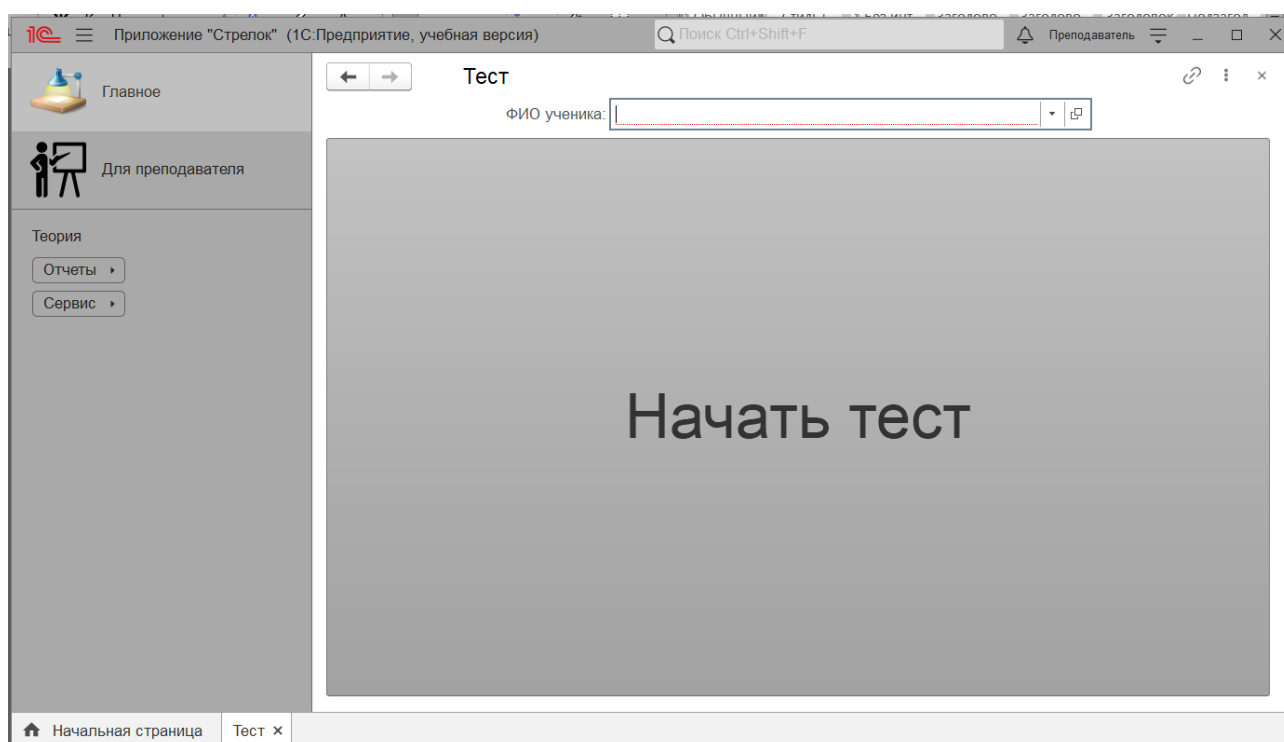


Рисунок 20 Окно теста под пользователем «Преподаватель»

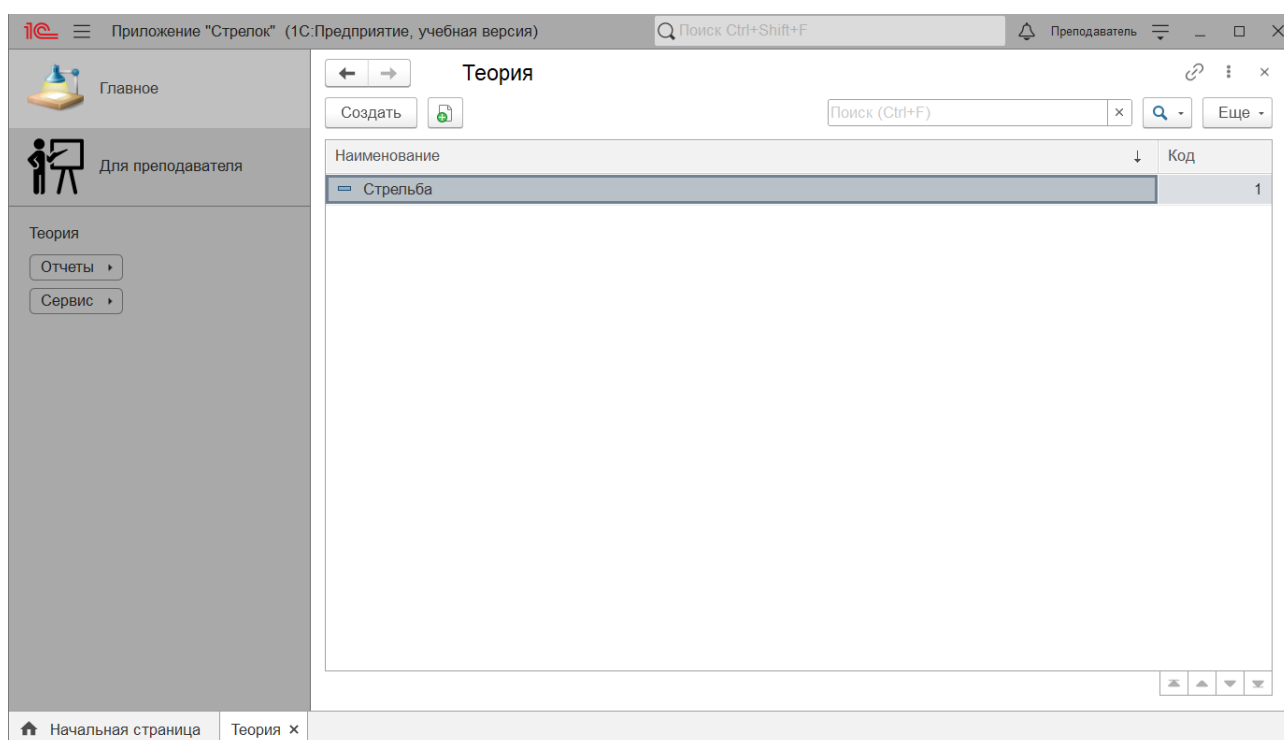


Рисунок 21 Окно справочника «Теория»

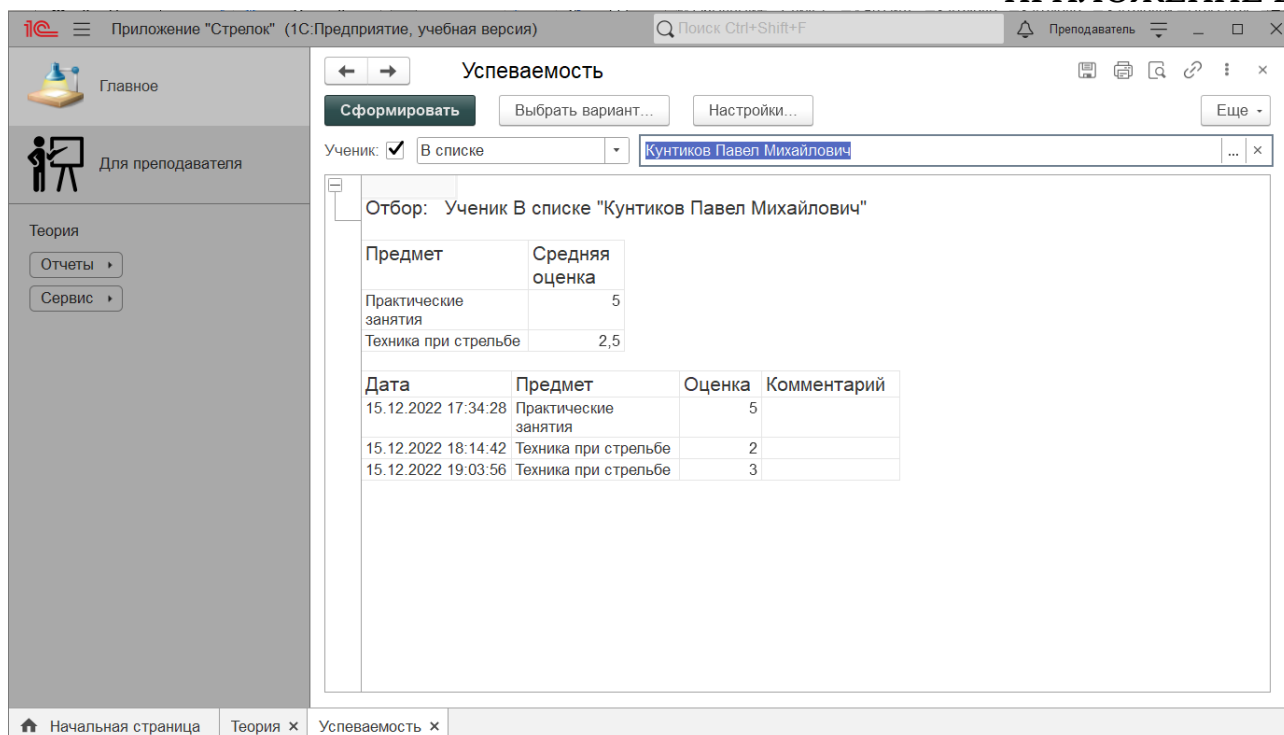


Рисунок 22 Окно отчета «Успеваемость»

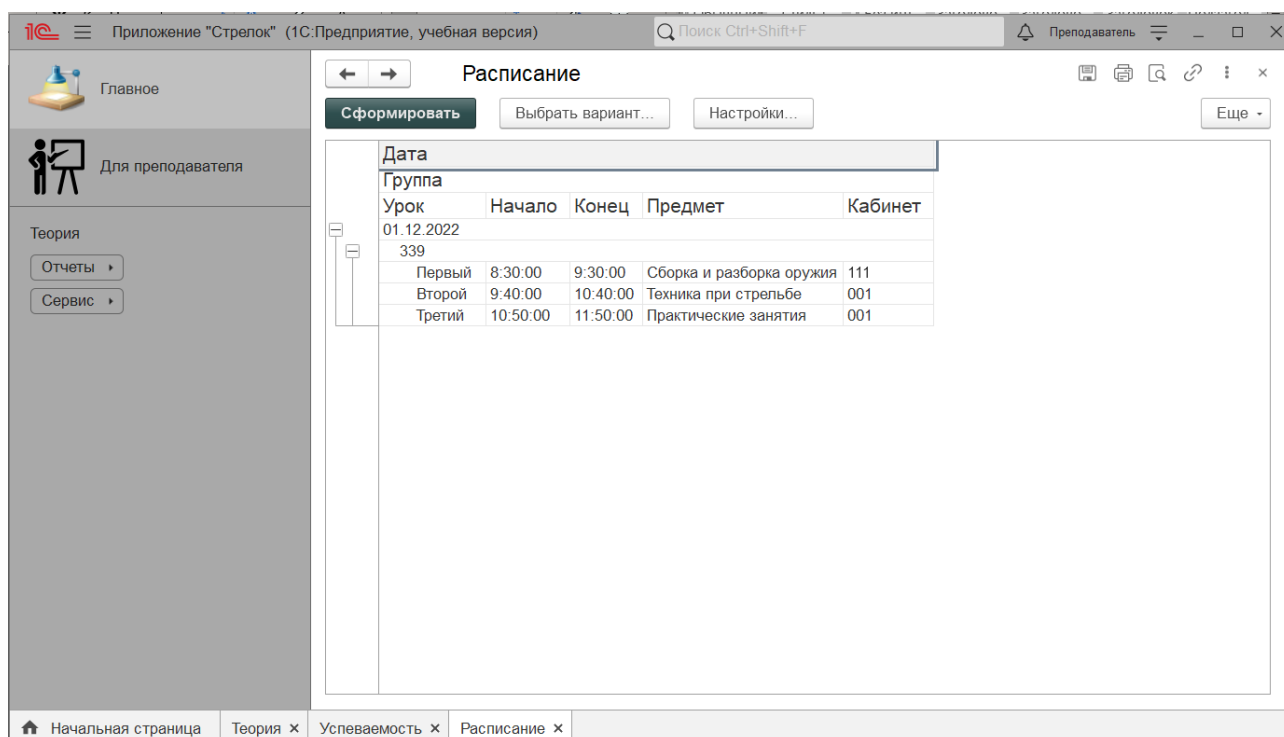


Рисунок 23 Окно отчета «Расписание»

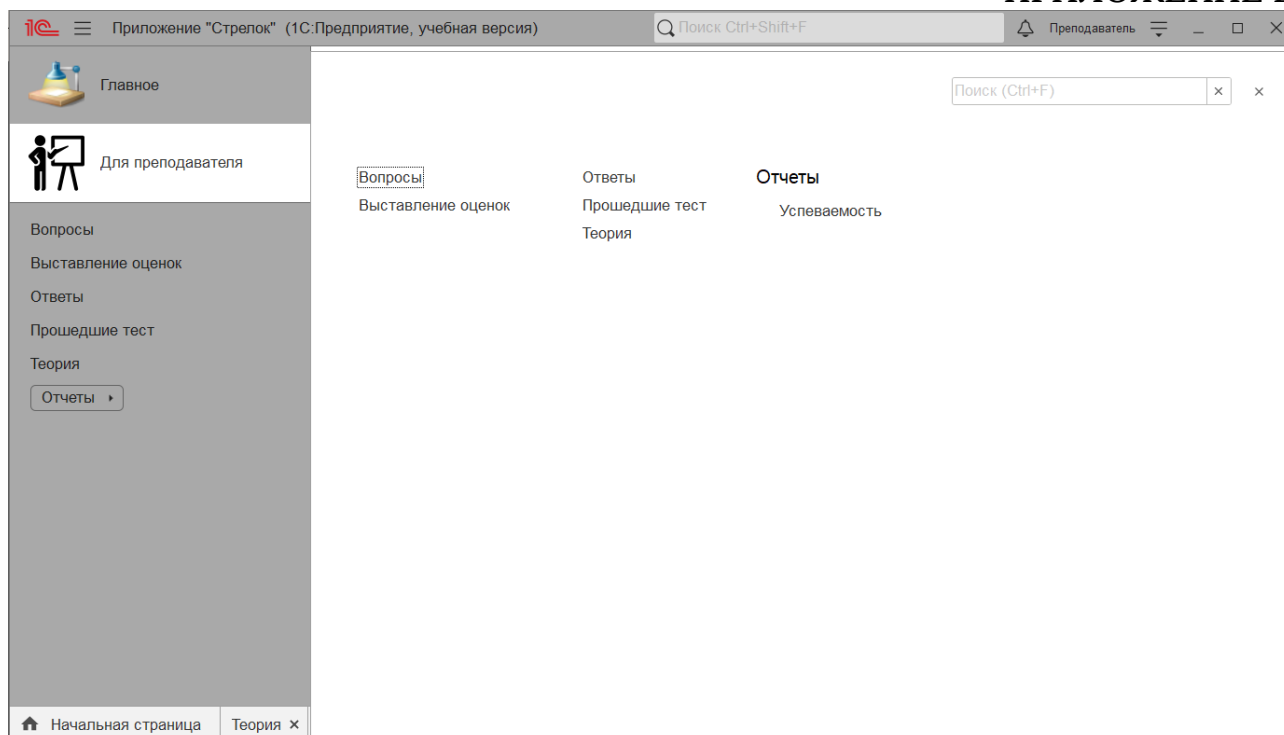


Рисунок 24 Окно подсистемы «Для преподавателя»

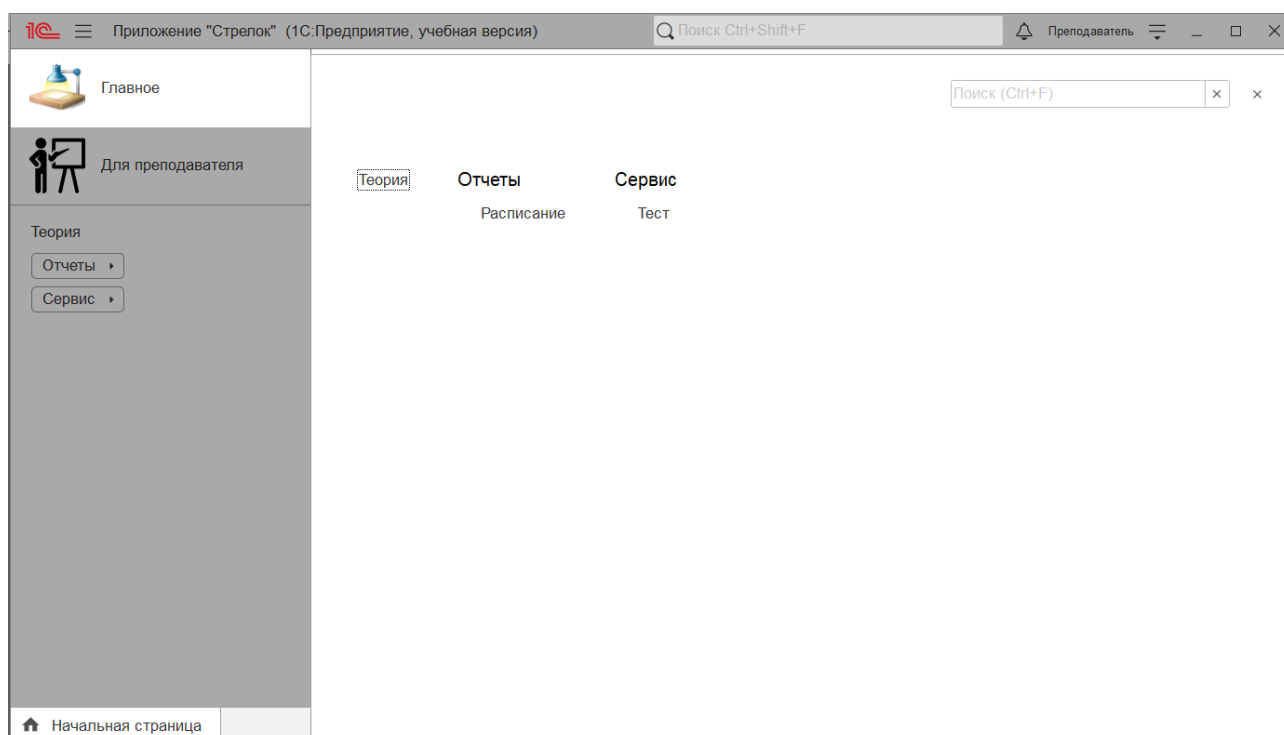


Рисунок 25 Окно встроенной подсистемы «Главное»

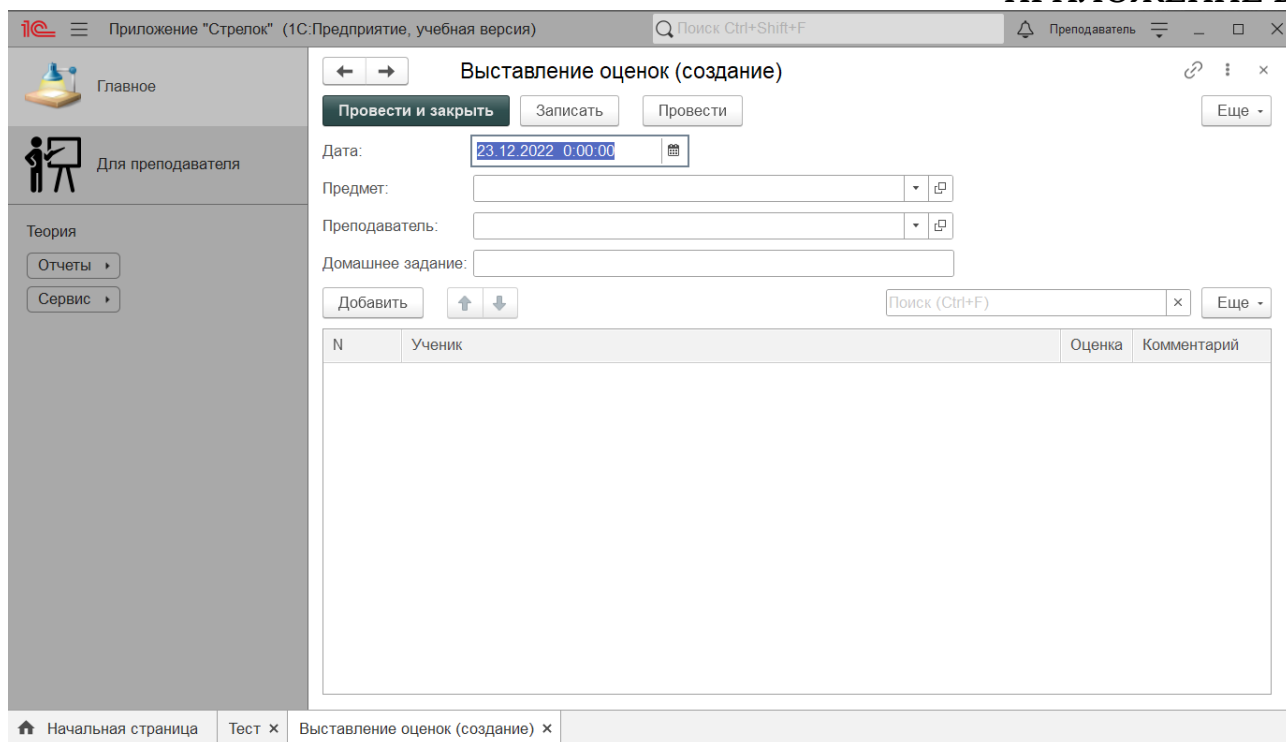


Рисунок 26 Окно добавления записи в документ «ВыставленияОценок»

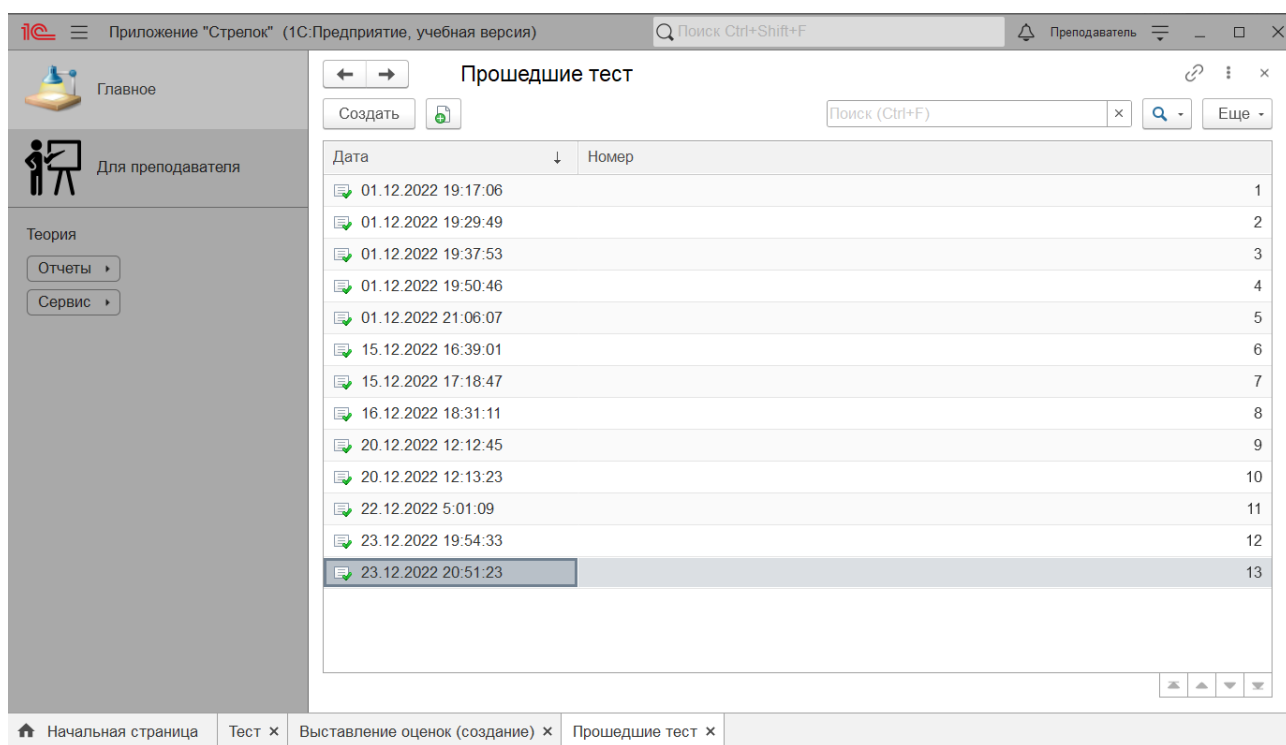


Рисунок 27 Окно списка документа «ПрошедшиеТест»

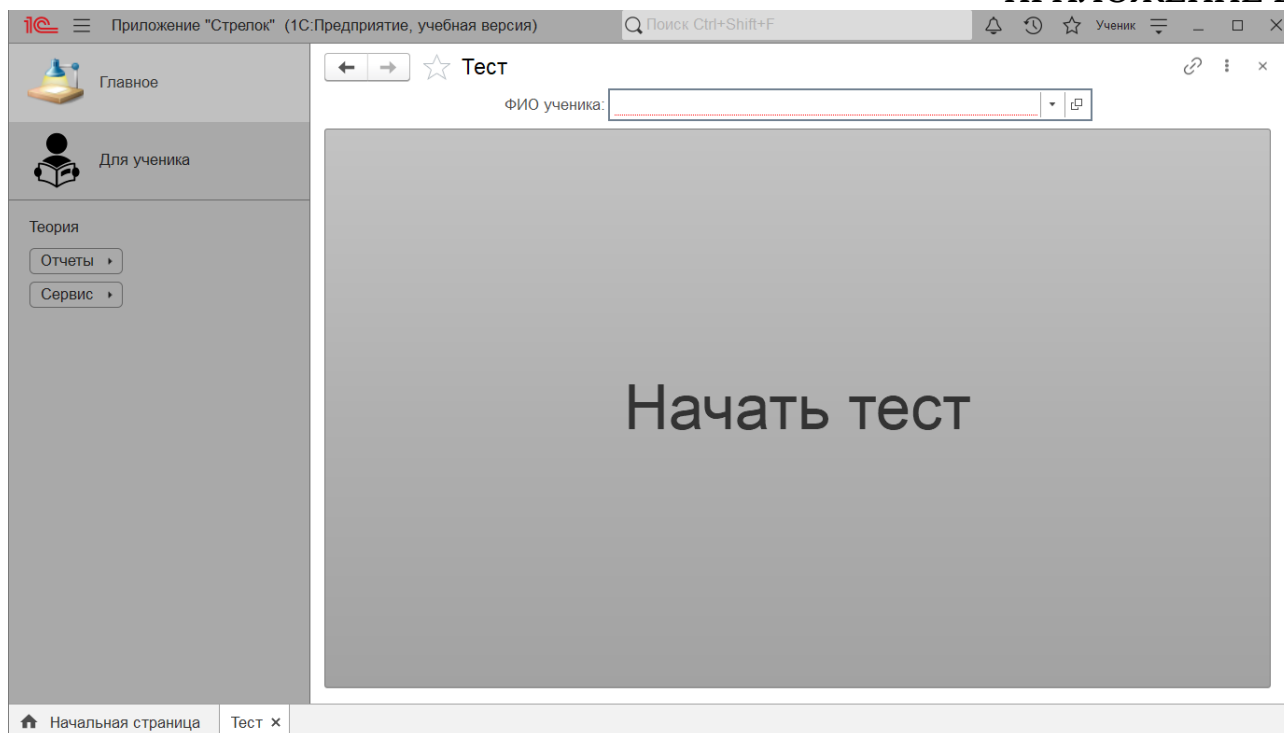


Рисунок 28 Окно теста под пользователем «Ученик»

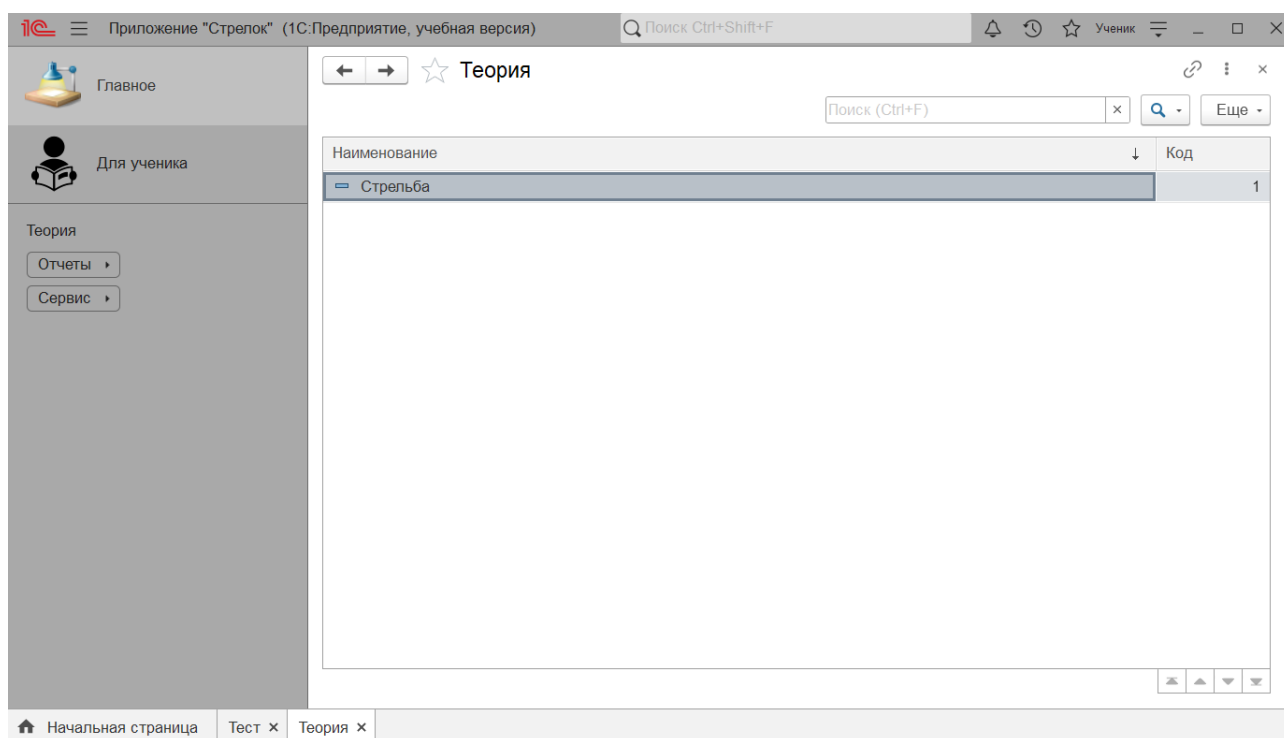


Рисунок 29 Окно списка справочника «Теория» под пользователем «Ученик»

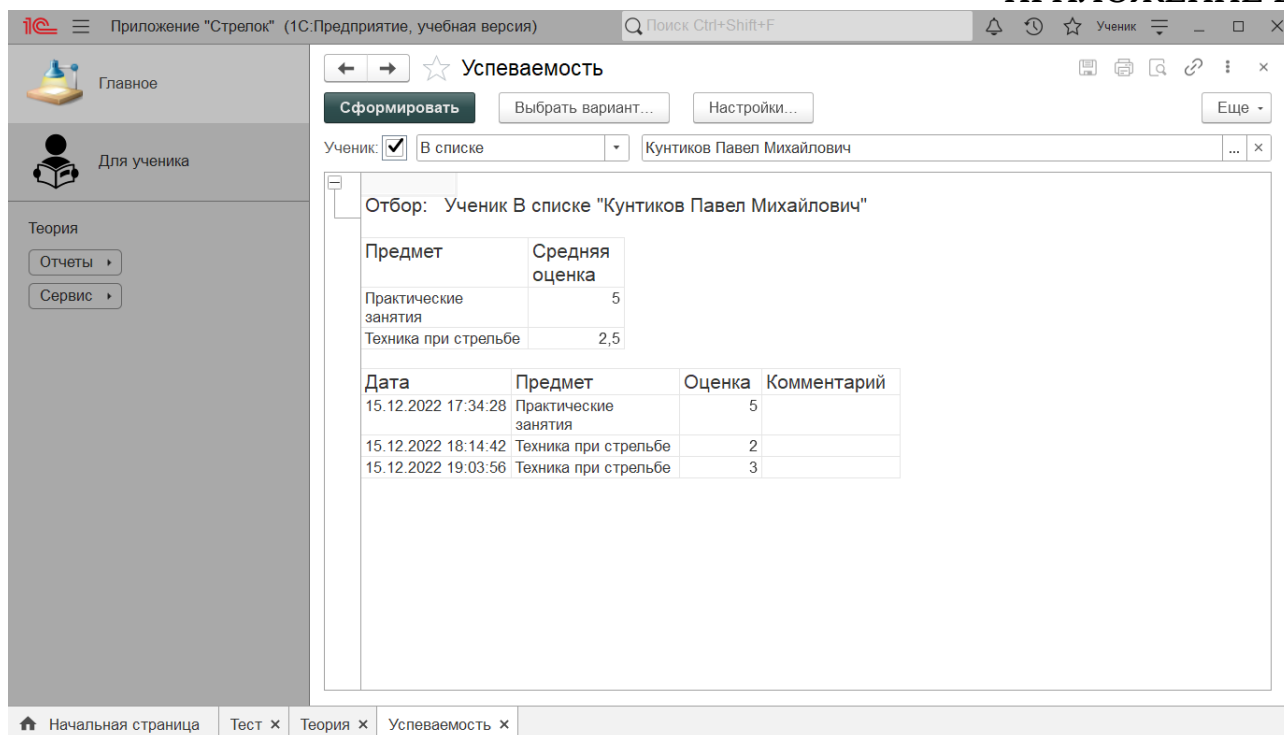


Рисунок 30 Окно отчета «Успеваемость» под пользователем «Ученик»

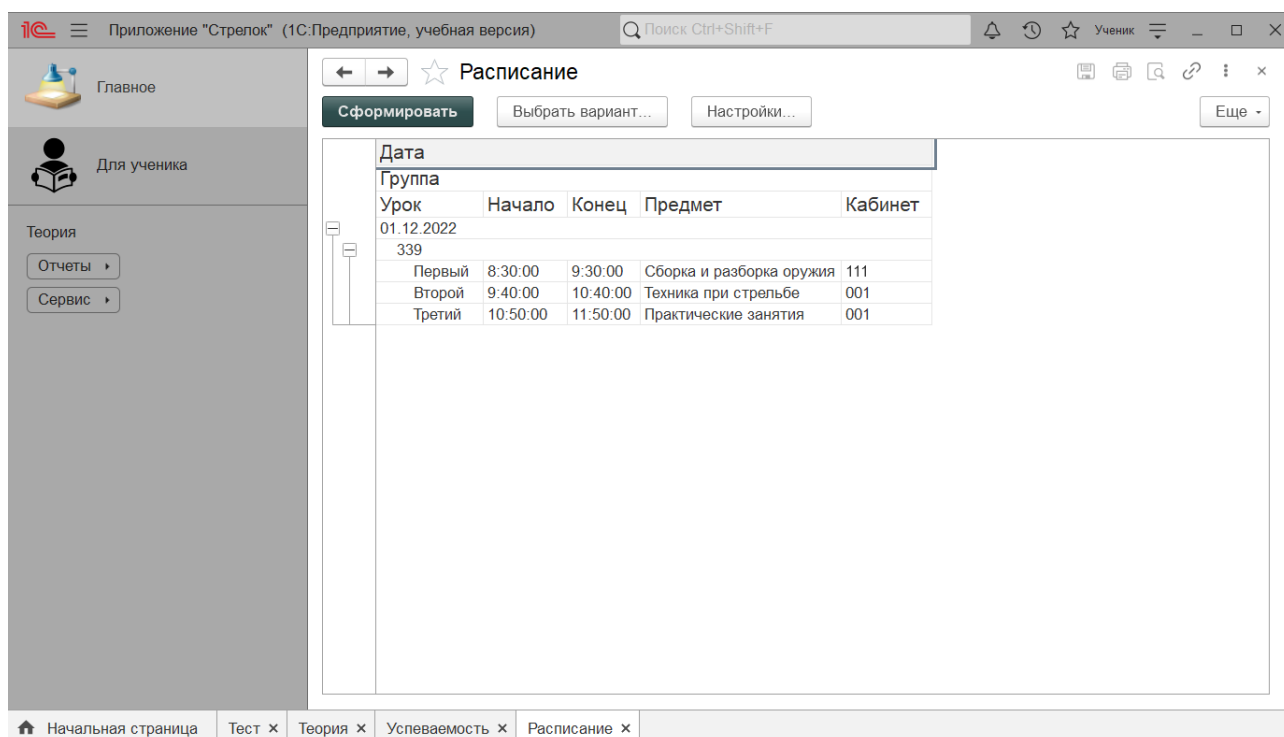


Рисунок 31 Окно отчета «Расписание» под пользователем «Ученик»

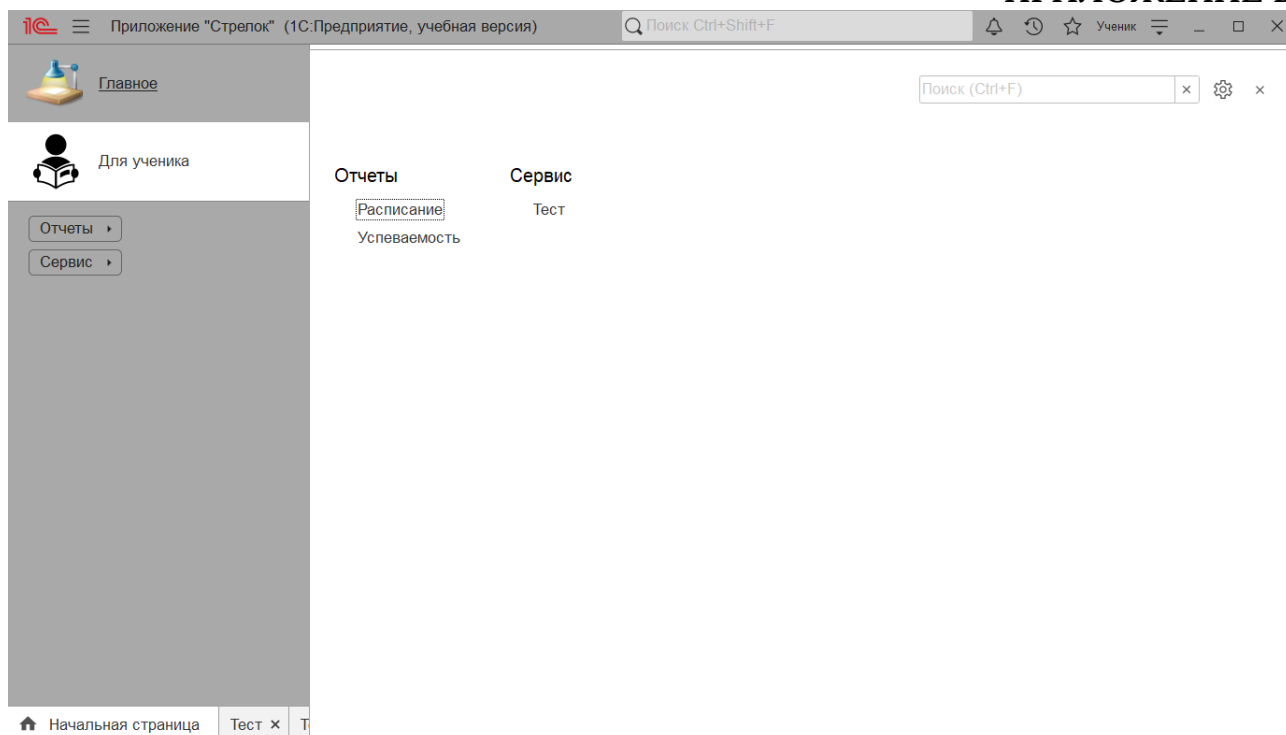


Рисунок 32 Окно подсистемы «Для ученика»

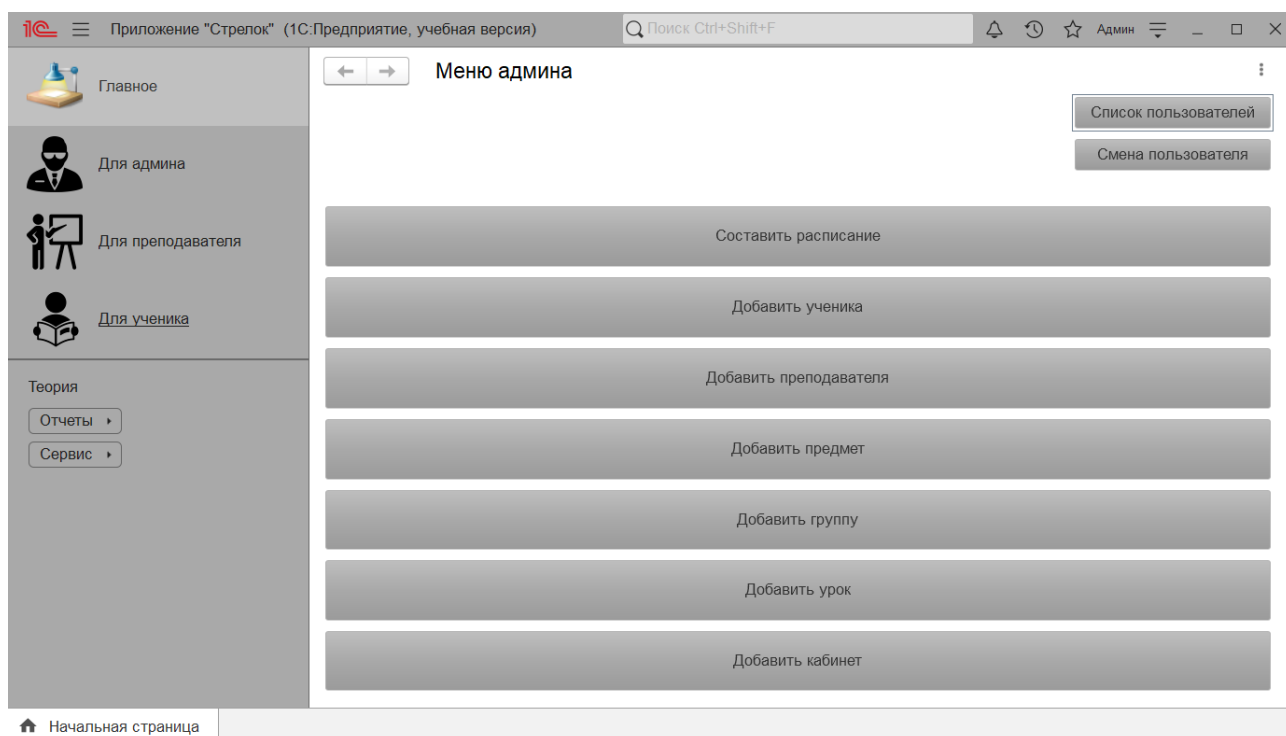


Рисунок 33 Окно админа

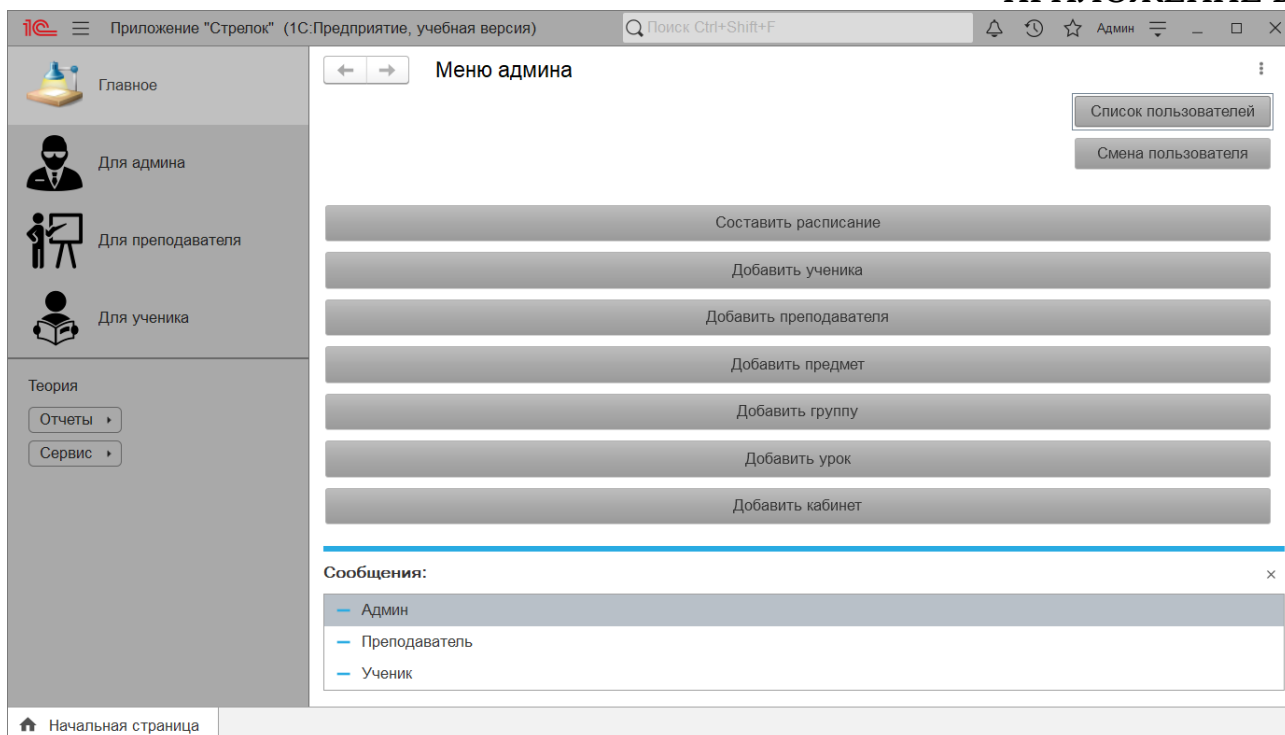


Рисунок 34 Работы функции «СписокПользователей»

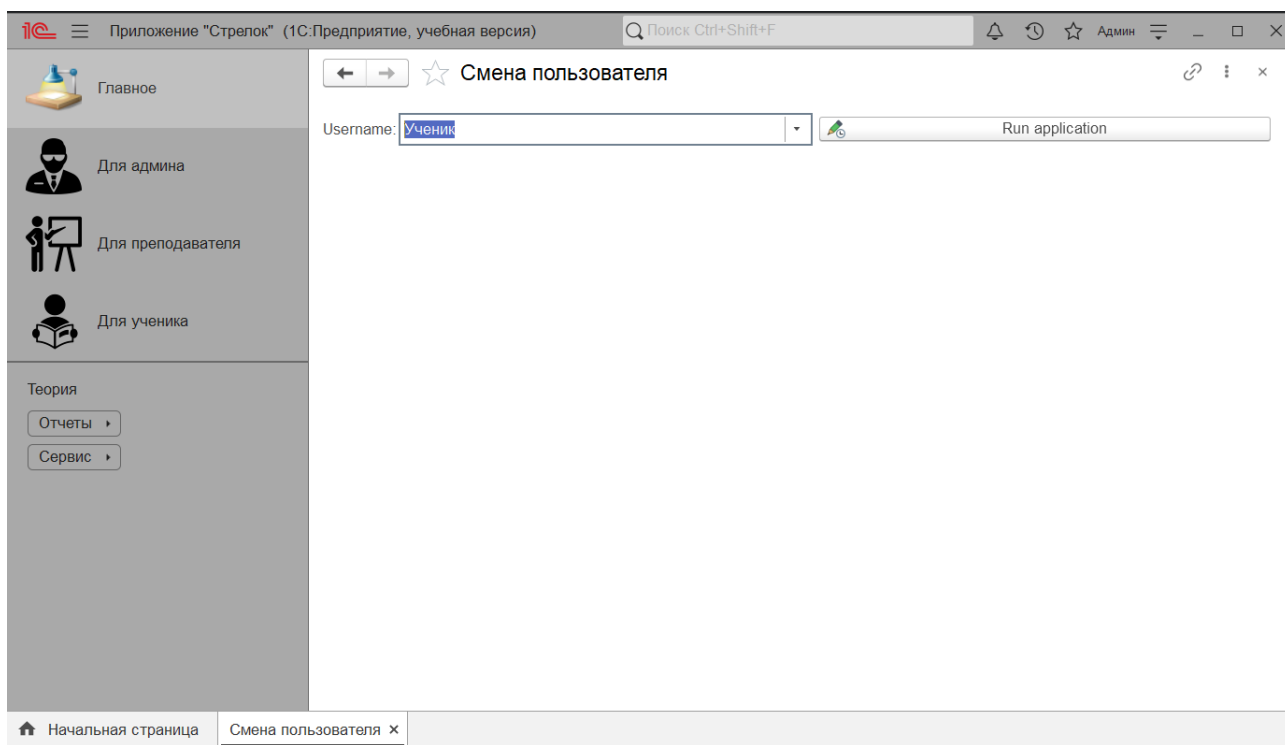


Рисунок 35 Окно смены пользователя под пользователем «Админ»

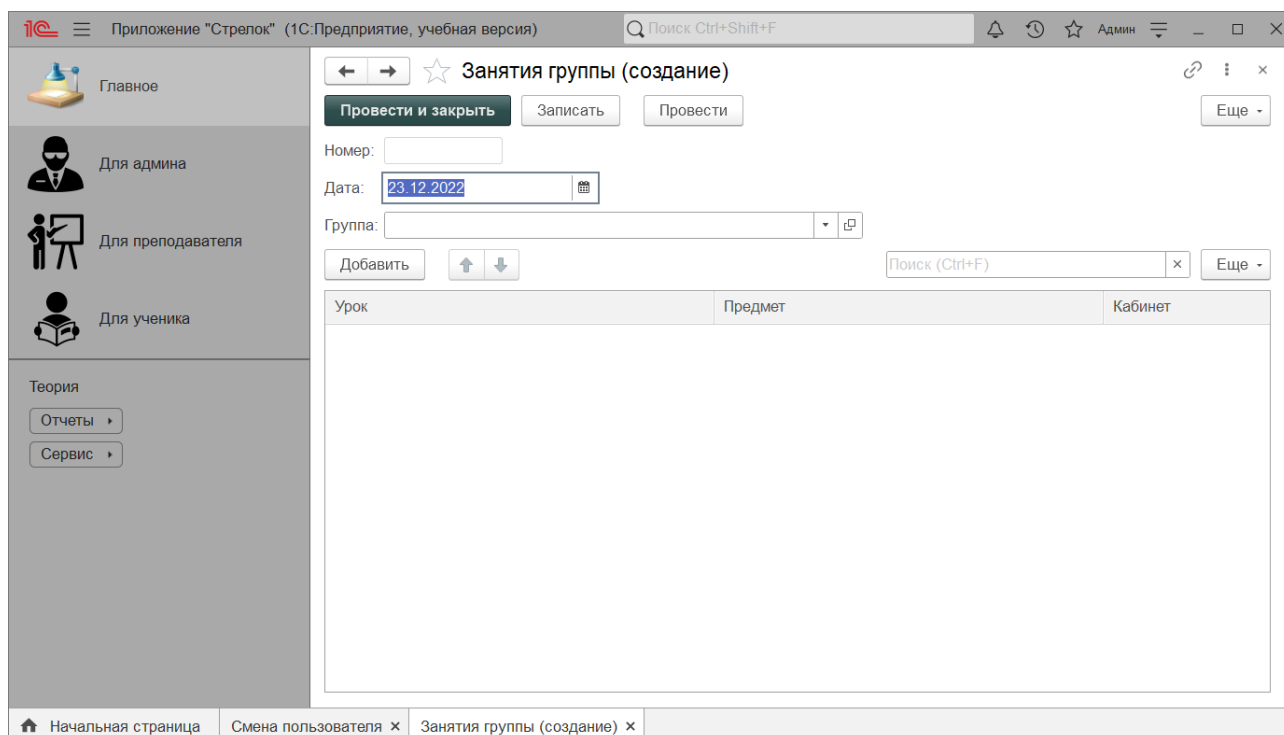


Рисунок 36 Окно документа «ЗанятияГруппы»

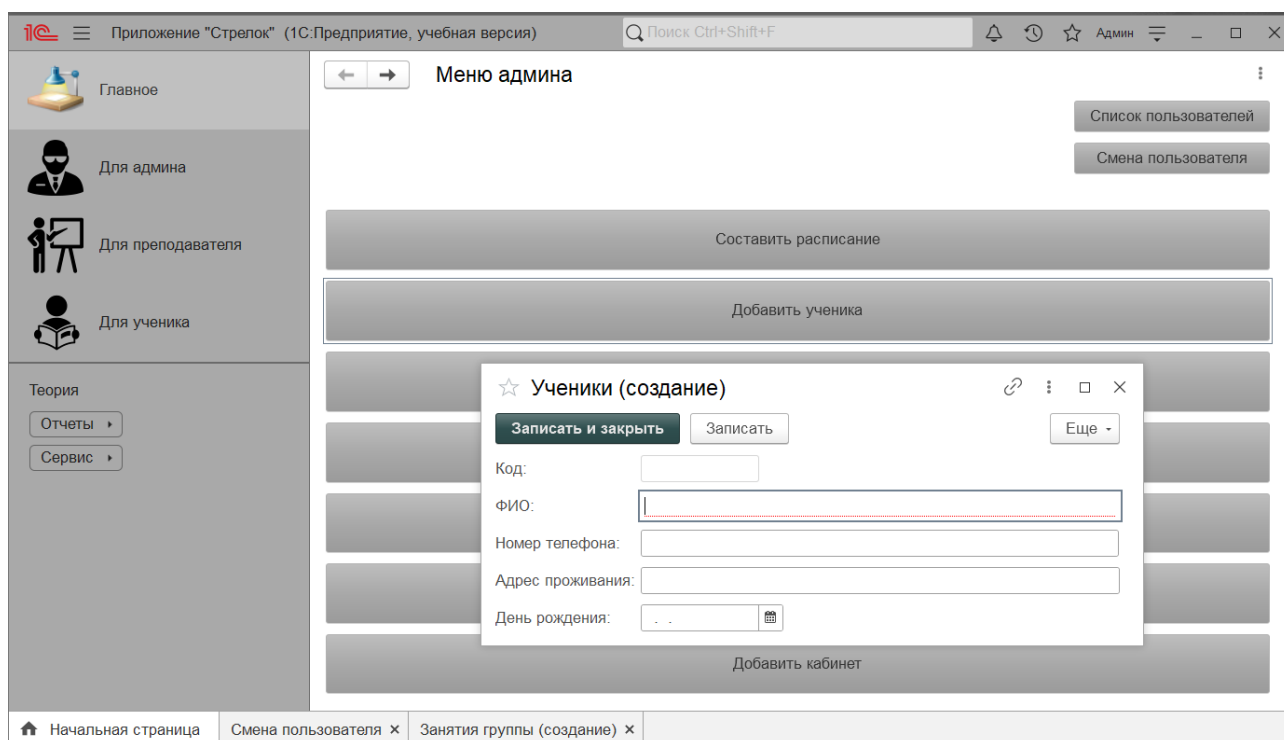


Рисунок 37 Окно добавления записи в справочник «Ученики»

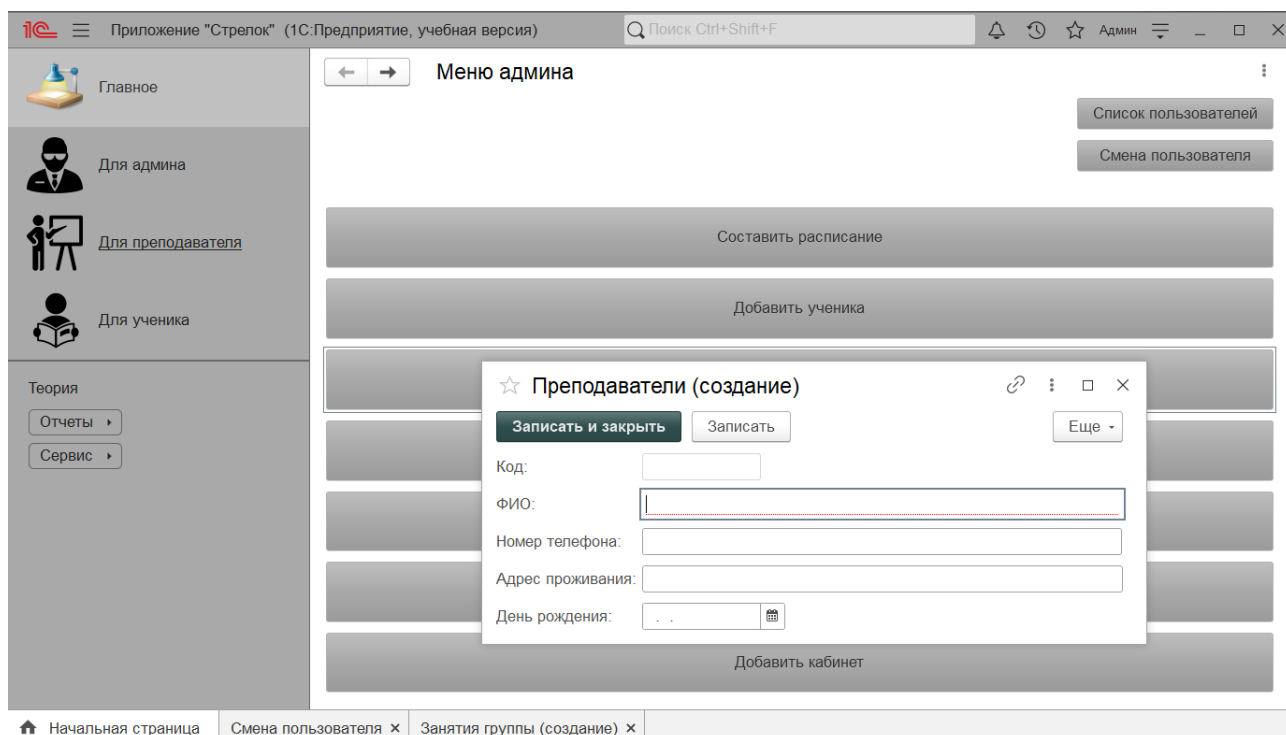


Рисунок 38 Окно добавления записи в справочник «Преподаватели»

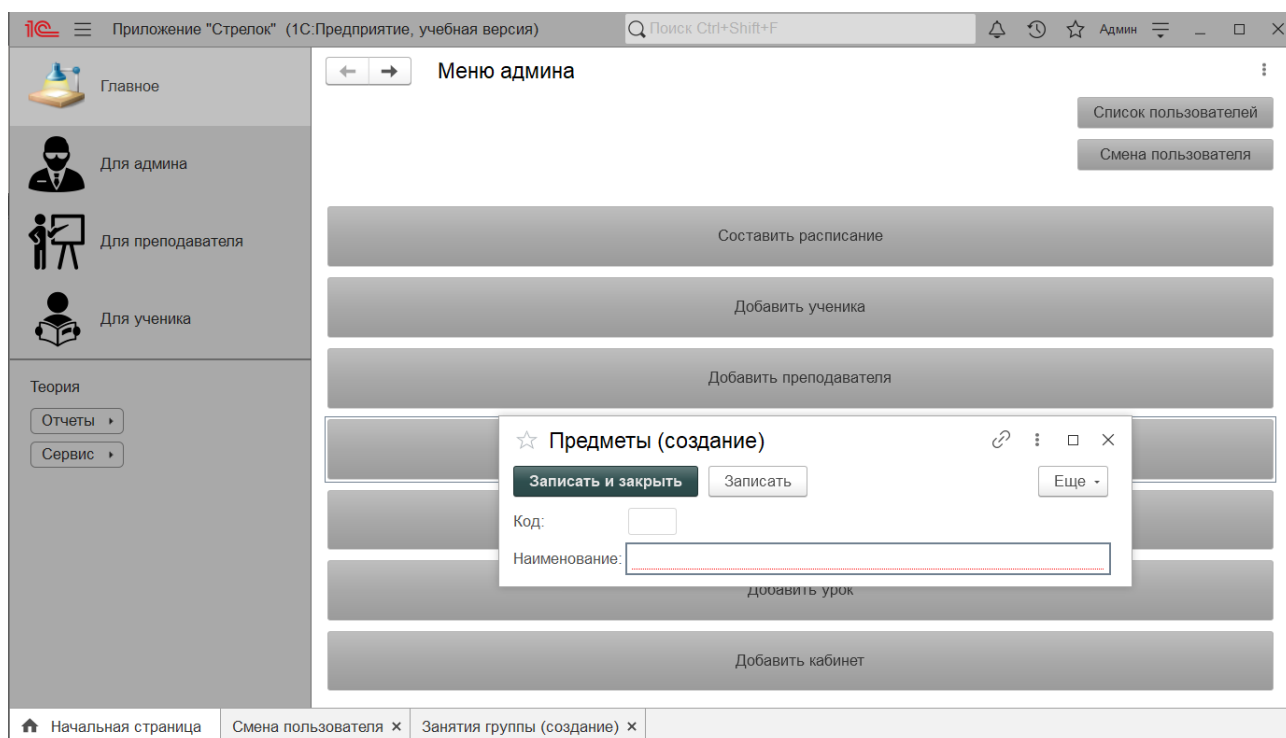


Рисунок 39 Окно добавления записи в справочник «Предметы»

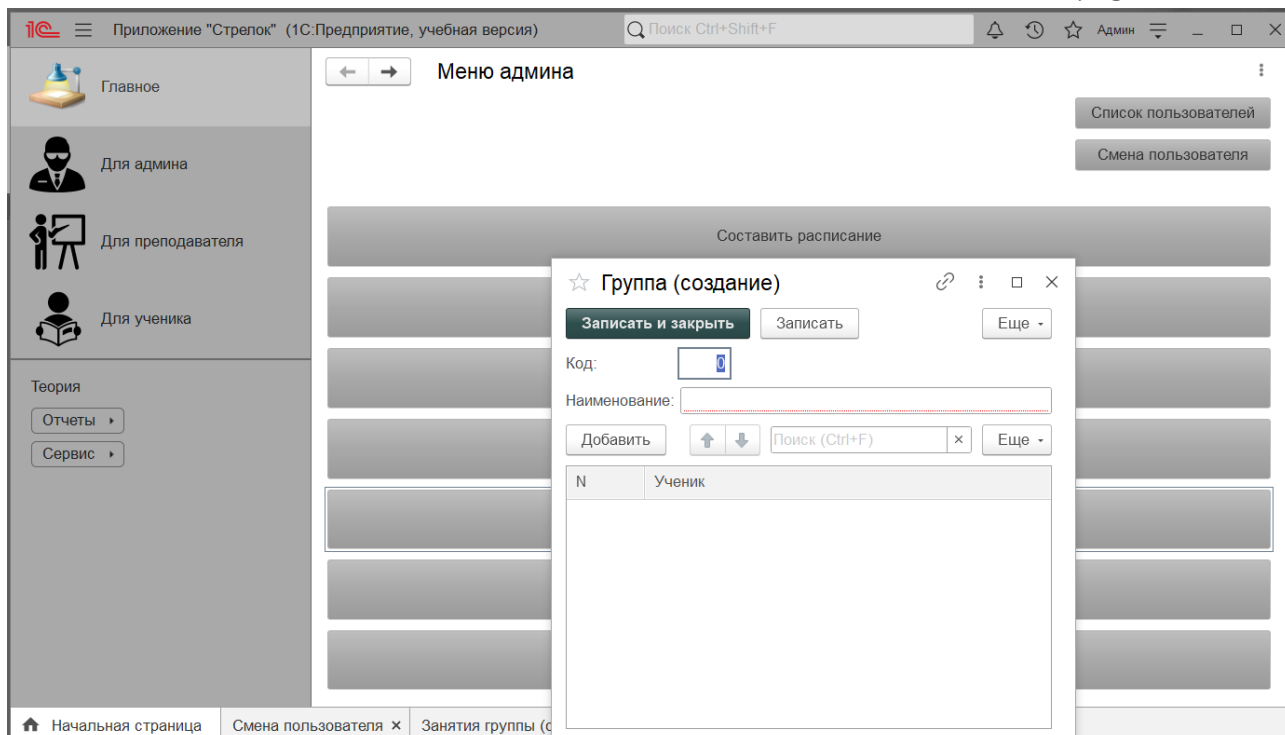


Рисунок 40 Окно добавления записи в справочника «Группа»

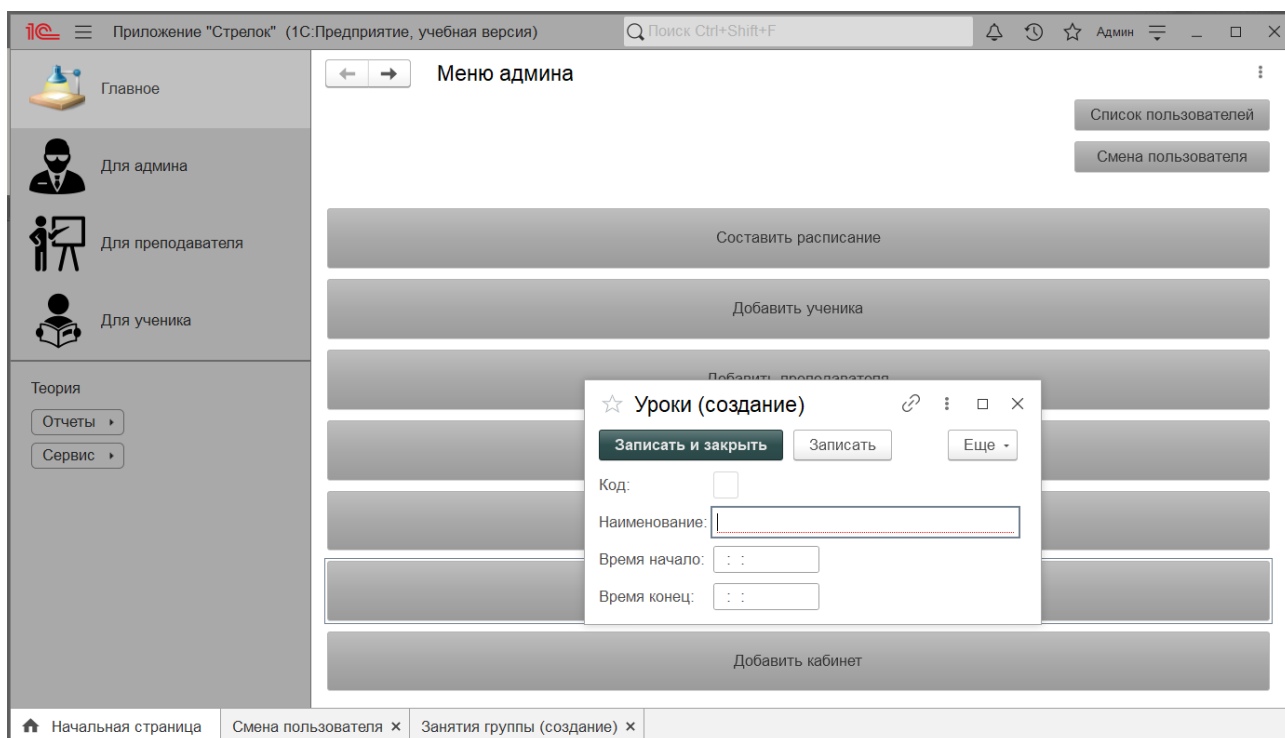


Рисунок 41 Окно добавления записи в справочник «Уроки»

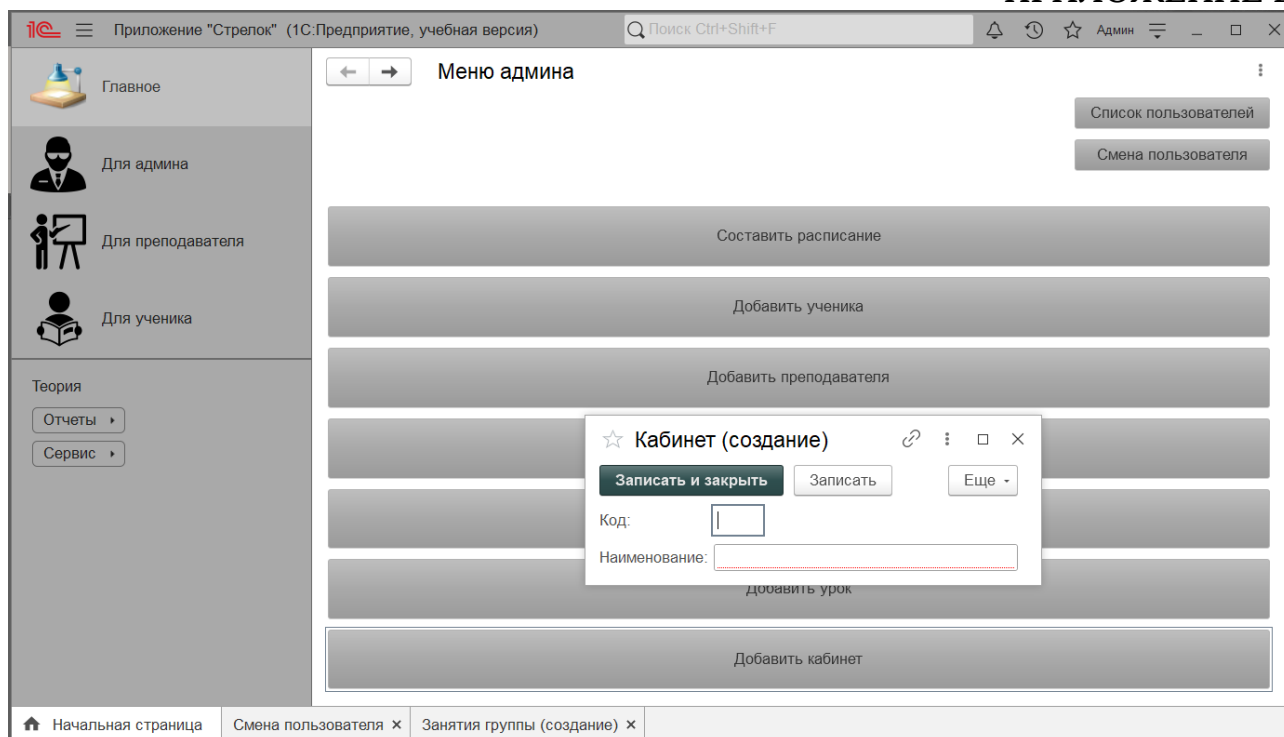


Рисунок 42 Окно добавления записи в справочник «Кабинет»

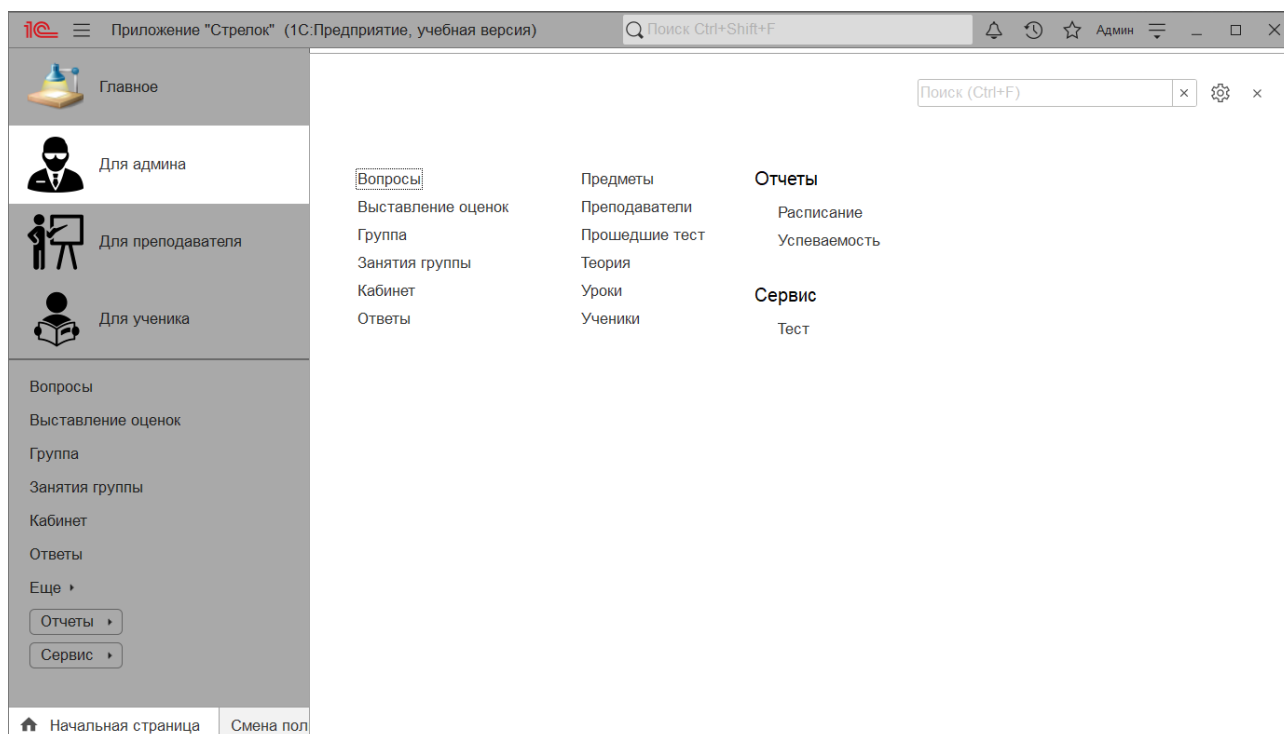


Рисунок 43 Окно подсистемы «Для админа» под пользователем «Админ»

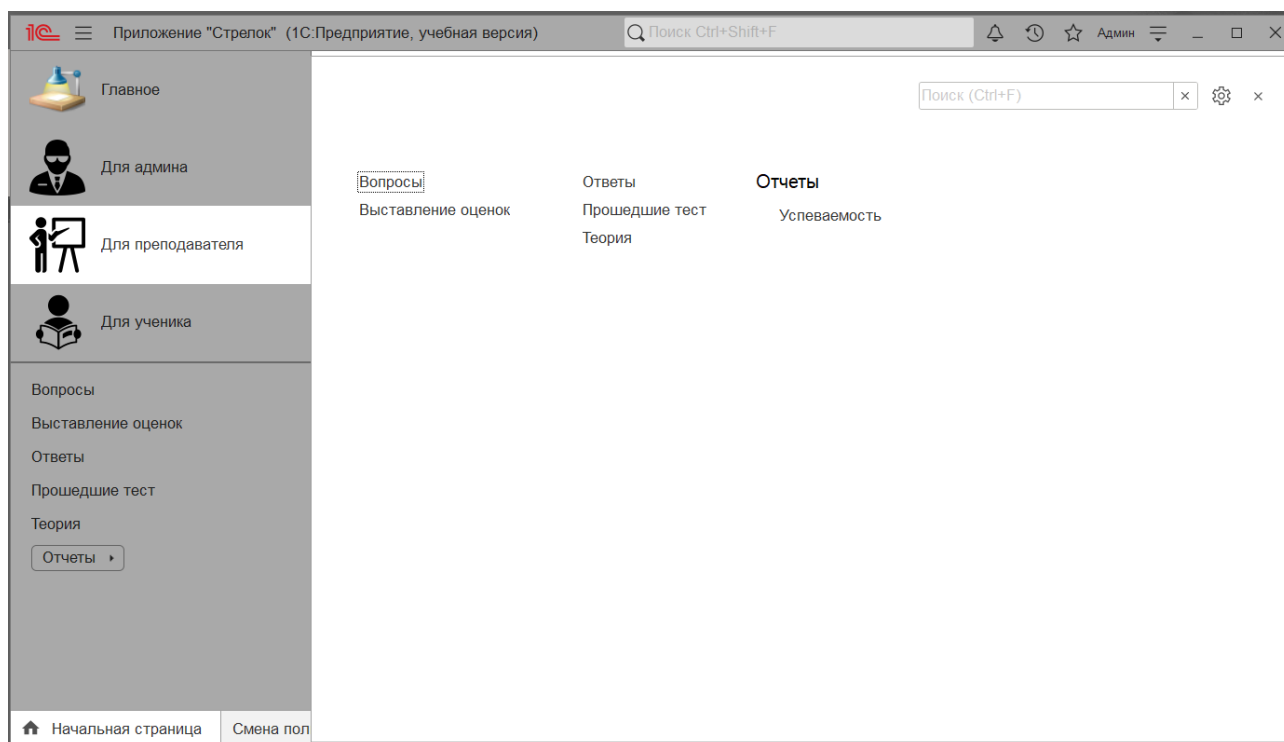


Рисунок 44 Окно подсистемы «Для преподавателя» под пользователем «Админ»

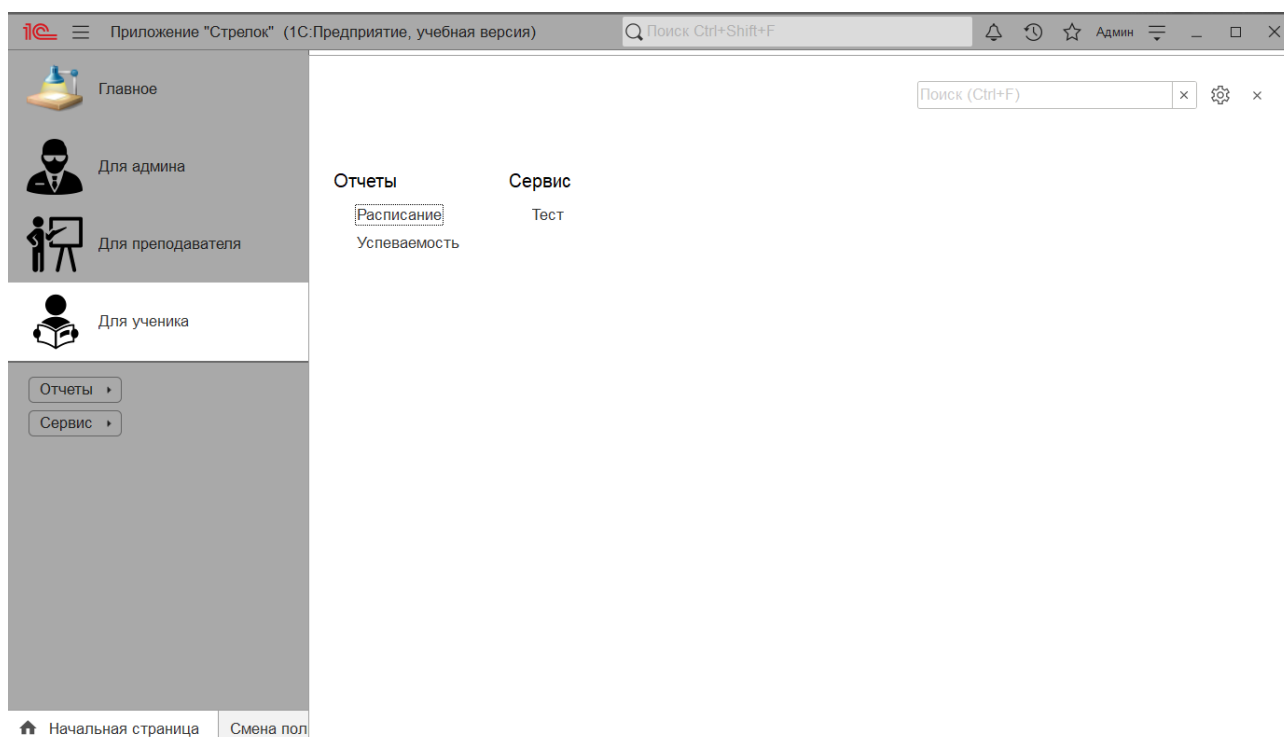


Рисунок 45 Окно подсистемы «Для ученика» под пользователем «Админ»

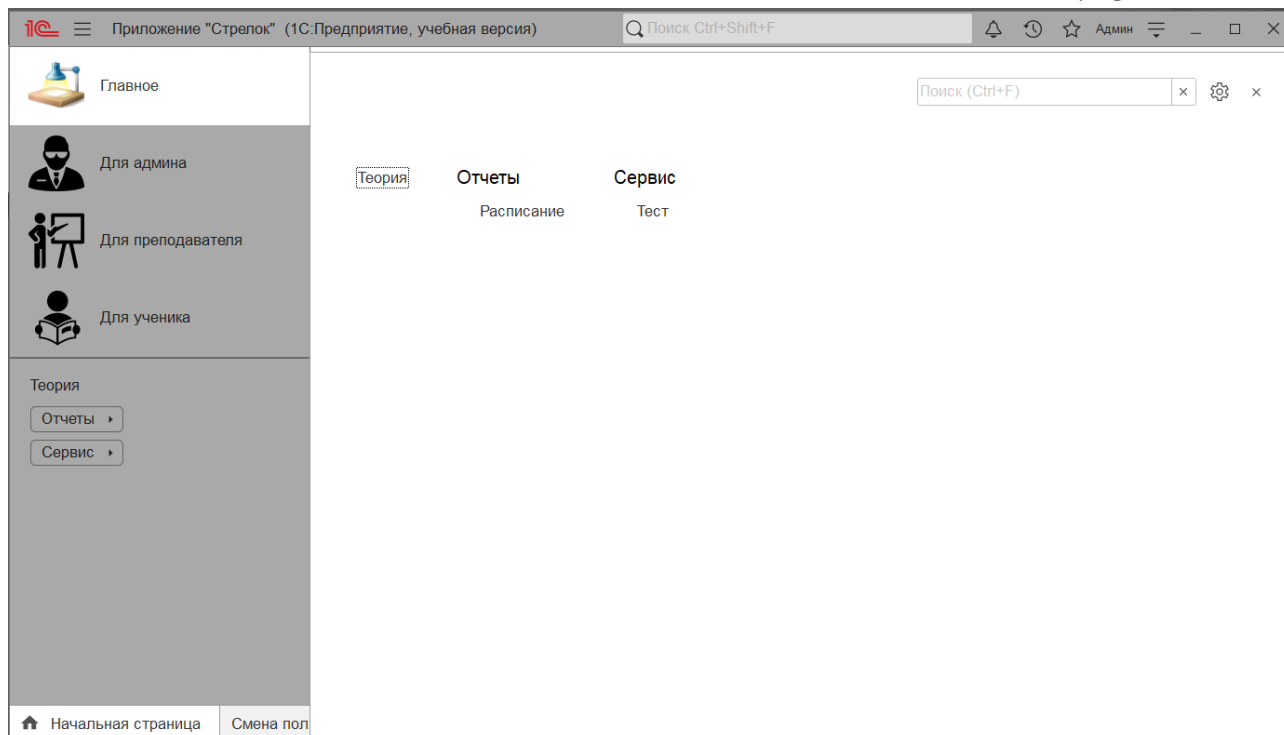


Рисунок 46 Окно встроенной подсистем «Главное» под пользователем «Админ»

3.3 Код модулей системы

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

Модуль общей формы «Меню Ученика»

&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

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

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                                     =
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

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

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

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;

3.Текст="

|ВЫБРАТЬ

| Ссылка

|ИЗ

| Справочник.Ответы

|ГДЕ Владелец = &Вопрос

|УПОРЯДОЧИТЬ ПО КОД";

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

Выб = 3.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

Модуль формы MainForm 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" , "change 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" , "Change 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));
```

```
    ElseIf 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;

    TrimmedText = Upper(TrimAll(Text));

    For Each InfoBaseUser In ArrayOfInfoBaseUsers Do

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

    ElseIf 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;

    ElseIf 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 GetMappedString(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)

 ChoiceDataRow = New ValueList;

 If IsBlankString(Text) Then

 If IsBlankString(SearchString) Then

 Return ChoiceDataRow;

 Else

 Text = SearchString

 EndIf;

 EndIf;

 Original = Lower(Text);

 OriginalToRu = GetMappedString(Original, False);

 OriginalToEn = GetMappedString(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;

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

Order = 2;

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

Order = 3;

EndIf;

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

```

        EndIf;

    EndDo;

EndDo;

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

    ChoiceDataRow.SortByPresentation(SortDirection.Asc);

    ChoiceData = New ValueList;

    K = 0;

    For Each Item In ChoiceDataRow Do

        ChoiceData.Add(Item.Value);

        K = K + 1;

        If K = 40 Then

            Break;

        EndIf;

    EndDo;

    Return ChoiceData;

EndFunction

```

&AtClient

```

Procedure SearchTreeOfFiles(TreeItems, ChoiceDataRow, 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;

```

```

        ElseIf Left(LowerName, LengthOriginalToEn) =

```

```

OriginalToEn Then

```

```

            Order = 2;

```

```

        ElseIf Left(LowerName, LengthOriginalToRu) =

```

```

OriginalToRu Then

```

```

            Order = 3;

```

```

            EndIf;

```

```

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

```

```

        "." + Name +

```

```

        "

```

```

"

```

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
```