минобрнауки россии

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт информационных технологий |
| наименование института (факультета) |
| Кафедра математического и программного обеспечения ЭВМ |
| наименование кафедры |

ОТЧЕТ

по Производственной практике (Технологической практике 2)

Листов \_\_\_\_

Студента Маркелова С. А.,

группы 1ПИб-02-3оп-22

Место прохождения практики:

ООО «Совкомбанк Технологии»

Студент \_\_\_\_\_\_\_\_\_\_\_\_

                 (подпись)

|  |  |
| --- | --- |
| Руководитель практики от кафедры | МПО ЭВМ |
| Сальникова О.С., к.филол.н., доцент | |
| Руководитель практики от предприятия, организации, учреждения  (Ф.И.О., должность) | |
| Раскин А.А., старший разработчик | |

2025 г.

Оглавление

[Введение 3](#_Toc202743187)

[Раздел 1. Общая характеристика базы практики 5](#_Toc202743188)

[Раздел 2. Организация производства 7](#_Toc202743189)

[Раздел 3. Структура и функции подразделения базы практики 9](#_Toc202743190)

[Раздел 4. Реализация индивидуального задания 11](#_Toc202743191)

[Раздел 5. Мероприятия по охране труда и технике безопасности 24](#_Toc202743192)

[Заключение 26](#_Toc202743193)

[Список литературы 27](#_Toc202743194)

[Приложение. Текст программы 30](#_Toc202743195)

# Введение

Производственная технологическая практика проходила в дистанционном режиме. Сроки прохождения практики: 16.06.2025-12.07.2025.

Целями производственной практики являются:

* повышение качества профессиональной подготовки студентов;
* закрепление полученных знаний по дисциплинам профессионального цикла;
* совершенствование навыков самостоятельной практической работы по проектированию элементов программного обеспечения.

Производственная практика предусматривает выполнение следующих задач:

* прохождение инструктажа по технике безопасности;
* получение индивидуального задания;
* поиск и анализ литературы по теме практики;
* выполнение практических задач, соответствующих индивидуальному заданию;
* оформление отчета о практике.

В ходе прохождения практики было получено задание: разработать виджет для ФИС-платформы, который позволит манипулировать данными конкретного типа (из таблиц базы данных).

ФИС-платформа – это программная платформа, ориентированная на автоматизацию и оптимизацию процессов в финансовом секторе. Ее возможности позволяют значительно ускорить разработку решений, снизить затраты на создание и поддержку систем, а также адаптировать бизнес-процессы под изменения в экономической среде.

Технологическая практика направлена на закрепление связи теоретического обучения с практической деятельностью, на повышение уровня практической подготовки бакалавров, обеспечивает прикладную направленность и адаптацию к рынку труда и производственным условиям.

Технологическая практика базируется на модулях: Программирование, Инструментальные средства программирования, Проектирование систем управления данными, Архитектура программно-информационных систем.

Результаты, полученные в процессе выполнения заданий практики должны стать основой для понимания таких модулей как Операционные системы, Информационные и компьютерные сети, Технология разработки программного обеспечения, а также для выполнения выпускной квалификационной работы.

# Раздел 1. Общая характеристика базы практики

ООО «Совкомбанк Технологии» – одна из ведущих российских компаний в области разработки IT-решений и предоставления технологических сервисов для финансового сектора. Компания является ключевым технологическим партнером Совкомбанка и входит в экосистему группы «Совкомбанк», одного из крупнейших частных банков России. Штаб-квартира компании расположена в Москве, а филиалы и подразделения присутствуют в различных регионах России, что позволяет обеспечивать оперативное взаимодействие с бизнес-подразделениями банка по всей стране [12].

В штате ООО «Совкомбанк Технологии» работают высококвалифицированные специалисты в области программирования, тестирования, анализа данных, проектного управления, информационной безопасности и других направлений IT. Основной задачей компании является разработка и сопровождение программных продуктов, интеграция и поддержка банковских систем, развитие цифровых сервисов для клиентов банка, а также внедрение инновационных технологий, включая решения на базе искусственного интеллекта и машинного обучения.

ООО «Совкомбанк Технологии» обеспечивает бесперебойную работу IT-инфраструктуры Совкомбанка и активно участвует в цифровой трансформации банка, повышая его технологическую устойчивость и конкурентоспособность на рынке финансовых услуг. Компания реализует проекты по созданию собственных программных платформ, автоматизации бизнес-процессов, разработке мобильных и web-приложений, а также интеграции с внешними партнерами [9].

Ключевыми преимуществами ООО «Совкомбанк Технологии» являются высокая степень экспертизы в разработке IT-решений для банковской сферы, гибкость в подходах к управлению проектами и использование современных методологий разработки программного обеспечения, таких как Agile и DevOps. Это позволяет компании быстро адаптироваться к меняющимся требованиям рынка и потребностям клиентов банка.

ООО «Совкомбанк Технологии» активно инвестирует в развитие персонала, внедряет современные подходы к обучению и повышению квалификации сотрудников, что способствует созданию мотивированной и профессиональной команды. Компания придерживается стратегии развития, направленной на укрепление позиций на российском рынке IT-услуг для финансового сектора, повышение качества и надежности предоставляемых сервисов, а также расширение спектра цифровых решений в интересах Совкомбанка и его клиентов [8].

# Раздел 2. Организация производства

ООО «Совкомбанк Технологии» организует производство своих IT-услуг на базе современных методологий разработки программного обеспечения, включая Agile (Scrum, Kanban) [18] и практики DevOps [17], которые позволяют наладить гибкое управление проектами, повысить скорость выпуска новых решений и обеспечить стабильное качество продуктов.

Технологический процесс разработки строится на принципах непрерывной интеграции и доставки (CI/CD), что позволяет быстро разрабатывать, тестировать и внедрять программные обновления в эксплуатацию. Все проекты компании проходят полный цикл: сбор и анализ требований от заказчика (бизнес-подразделений Совкомбанка), проектирование архитектуры решений, разработка программного кода, модульное и интеграционное тестирование, внедрение в промышленную среду, а также последующая техническая поддержка [15].

Для управления проектами в компании применяются гибкие методики с регулярными спринтами и планированием задач, что обеспечивает прозрачность процессов для всех участников, быструю реакцию на изменения требований и эффективное взаимодействие между командами разработчиков, аналитиков и менеджеров. В качестве основных инструментов управления задачами используется система Redmine, которая позволяет вести учет всех задач, контролировать сроки их выполнения и отслеживать прогресс по проектам [20]. Для коммуникации внутри команд и проведения онлайн-встреч активно применяется сервис видеоконференций МТС Линк, который обеспечивает удобное и стабильное проведение созвонов как между сотрудниками внутри компании, так и с внешними партнерами [11].

Для контроля качества продукции используется автоматизированное тестирование, включающее юнит-тесты, функциональное и нагрузочное тестирование. Разработка и тестирование программных продуктов осуществляется с применением систем контроля версий (Git), что позволяет организовать параллельную работу нескольких команд и минимизировать риски ошибок при слиянии изменений в коде.

В части методов управления базой практики в АО «Совкомбанк Технологии» применяется модель матричной структуры управления, при которой сотрудники объединяются в кросс-функциональные команды под конкретные проекты, сохраняя подчинение своим профильным руководителям. Это повышает гибкость и скорость реализации проектов, а также позволяет эффективно распределять ресурсы между различными задачами.

Кроме того, в компании развита система внутреннего обучения и наставничества, что способствует быстрому вовлечению новых сотрудников и студентов-практикантов в процессы разработки, а также передаче знаний и опыта от старших специалистов к младшим [14].

# Раздел 3. Структура и функции подразделения базы практики

Практика в ООО «Совкомбанк Технологии» проходила в Отделе разработки залогового кредитовая и страхования.

Отдел разработки залогового кредитования и страхования (ОРЗКиС) занимается созданием, развитием и сопровождением программных продуктов, обеспечивающих автоматизацию процессов, связанных с залоговым кредитованием и страховыми услугами [12].

Структура подразделения включает несколько ключевых ролей, каждая из которых выполняет определённые функции для обеспечения бесперебойной работы и развития программных решений:

* руководитель отдела – осуществляет общее управление, планирование работы, распределение задач между сотрудниками, контроль сроков и качества выполнения проектов;
* тимлиды (ведущие разработчики) – отвечают за архитектуру решений, постановку технических задач, взаимодействие с аналитиками и заказчиками, координацию работы команды разработчиков;
* разработчики программного обеспечения – занимаются непосредственной реализацией программных модулей, исправлением ошибок, оптимизацией существующих решений, написанием автоматизированных тестов;
* системные аналитики – анализируют бизнес-процессы, формируют требования, составляют технические задания для команды разработки, взаимодействуют с заказчиками и другими подразделениями;
* тестировщики – проводят ручное и автоматизированное тестирование новых функций и исправлений, формируют баг-репорты, проверяют соответствие реализованных решений требованиям;
* сопровождающий персонал – специалисты по технической поддержке и DevOps, которые отвечают за развертывание решений на серверах, поддержку инфраструктуры, настройку окружений, мониторинг стабильности систем.

Подразделение ОРЗКиС выполняет следующие функции:

* разработка, внедрение и сопровождение программных продуктов для автоматизации процессов залогового кредитования и страхования;
* поддержка и развитие существующих программных решений, оперативное исправление обнаруженных ошибок и багов;
* анализ требований заказчиков, моделирование бизнес-процессов и проектирование архитектуры программных систем;
* интеграция программных решений с внутренними и внешними сервисами, такими как CRM, бухгалтерские системы, страховые сервисы;
* обеспечение безопасности и отказоустойчивости разрабатываемых решений, в том числе соответствие стандартам информационной безопасности;
* проведение тестирования программных продуктов, обеспечение качества выпускаемых решений;
* подготовка и сопровождение технической документации по разрабатываемым и поддерживаемым системам.

# Раздел 4. Реализация индивидуального задания

В качестве задания на практику было необходимо разработать виджет для ФИС-платформы, который позволит манипулировать данными конкретного типа (из таблиц базы данных).

Разработка виджета велась в несколько этапов.

1 этап – знакомство с предметной областью проекта.

Разработка виджета ведется на ФИС-платформе. ФИС-платформа – это программная платформа, ориентированная на автоматизацию и оптимизацию процессов в финансовом секторе. Она основана на использовании low-code подхода [7].

Low-code (в переводе с англ. «мало кода») – это подход к разработке программного обеспечения, в котором большая часть работы выполняется с помощью визуальных инструментов [4]. Разработчики используют визуальные инструменты, такие как конструкторы интерфейсов и процессов, вместо того чтобы вручную писать программный код. Этот метод ускоряет создание программных продуктов и снижает требования к квалификации разработчиков.

Основные характеристики ФИС-платформы:

1. Инструменты для визуальной разработки – платформа предлагает инструменты, которые позволяют проектировать интерфейсы и бизнес-процессы без необходимости программирования. Например:

* конструктор форм – визуальный инструмент для создания пользовательских интерфейсов;
* дизайнер процессов – инструмент, который позволяет «нарисовать» алгоритм работы системы (например, последовательность обработки заявки);

1. Готовые шаблоны решений – платформа включает набор типовых шаблонов, которые можно адаптировать для конкретных задач. Это ускоряет внедрение системы и снижает затраты на разработку с нуля;
2. Интеграция с другими системами – платформа поддерживает обмен данными с другими информационными системами через стандартные протоколы, такие как REST API (интерфейсы для взаимодействия с другими приложениями) и SOAP (протокол для передачи данных в формате XML);
3. Масштабируемость и высокая производительность – платформа подходит для работы с большими объемами данных и высоким количеством пользователей, что важно для банков и крупных финансовых организаций.

2 этап – изучение проектной документации.

В рамках данного этапа была изучена следующая документация по ФИС-платформе [13]:

* Руководство пользователя (рис. 1);
* Справочник функций (рис. 2);
* Руководство администратора (рис. 3);
* Начало работы в Platform (рис. 4).

Также был просмотрен видеокурс от разработчиков ФИС-платформы «Учимся работать с платформой» [16].

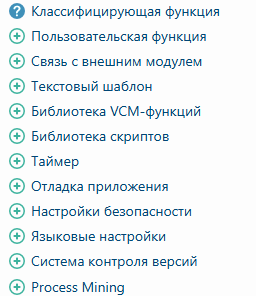
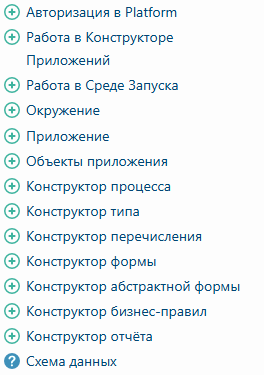


Рис. 1. Подразделы Руководства пользователя

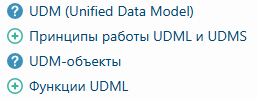


Рис. 2. Подразделы Справочника функций

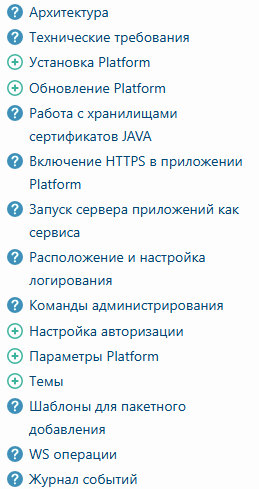
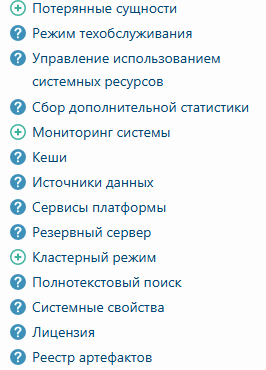
 

Рис. 3. Подразделы Руководства администратора

Изображение выглядит как текст, Шрифт, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 4. Подразделы Работы в Platform

3 этап – анализ существующих виджетов на ФИС-платформе.

Заказчик помимо технического задания также предоставил набор уже существующих виджетов (рис. 5). На основе предоставленных материалов был проведён анализ текущих решений, что позволило использовать полученные данные для проектирования нового виджета, соответствующего требованиям заказчика.

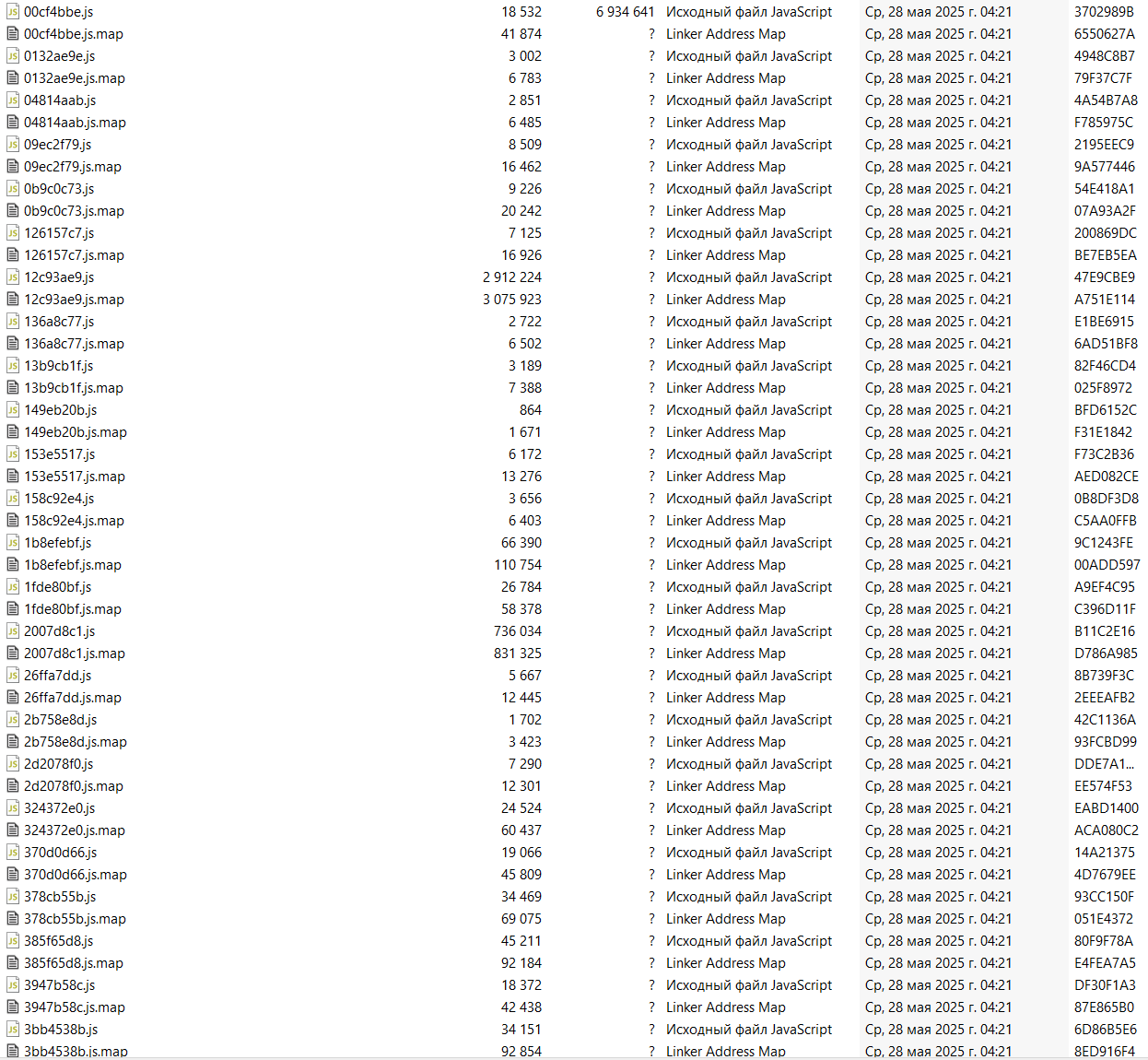


Рис. 5. Набор существующих виджетов

4 этап – анализ требований и постановка задач.

Заказчиком была поставлена задача разработать виджет ФИС-платформы, который отвечает следующим требованиям:

* виджет должен позволять манипулировать данными из конкретного типа (из таблиц базы данных);
* виджет должен представлять собой самодостаточный элемент интерфейса, который можно использовать любое количество раз;
* виджет должен иметь конечный набор свойств и событий (функций/методов);
* виджет должен принимать на вход данные, описывающие то, что пользователь хочет увидеть на экране;
* интерфейс элементов виджета должен соответствовать макетам (рис. 6-9).

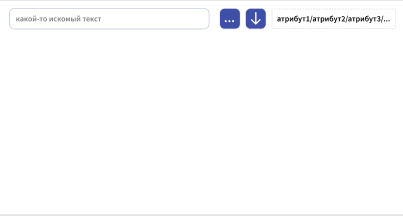


Рис. 6. Макет окна поиска

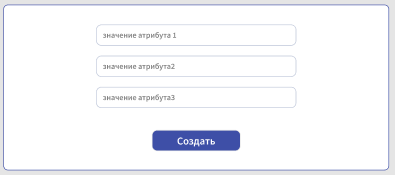


Рис. 7. Макет окна создания записей

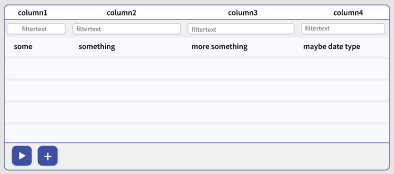


Рис. 8. Макет окна фильтрации

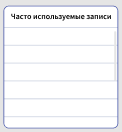


Рис. 9. Макет окна часто используемых записей

Разработка виджета велась в проектной команде из нескольких человек. Основные задачи, которые было необходимо решить в рамках разработки виджета:

* реализация базовой логики и инициализации виджета;
* реализация UI и генерации HTML;
* реализация обработки событий и динамики.

Данные задачи были поделены между участниками проектной команды. В качестве индивидуального вклада в проект была поставлена задача реализовать базовую логику виджета и методы его инициализации.

Для достижения целей были поставлены следующие подзадачи:

* работа с параметрами и настройками виджета;
* реализация истории изменений;
* реализация методов init, addToHistory, геттеров и сеттеров;
* поддержка мок-данных и поиска.

5 этап – выбор технологий и инструментов.

Проект реализуется на ФИС-платформе. Она предоставляет low-code инструменты для разработки интерфейсов и управления бизнес-процессами [19].

Для разработки функционала виджетов, как элементов интерфейса, и их интеграции с системами управления, был выбран язык программирования JavaScript [6]. Это высокоуровневый язык программирования, который используется для создания динамического и интерактивного поведения на web-страницах. Он позволяет изменять содержимое страницы, реагировать на действия пользователя, работать с сервером без перезагрузки страницы и создавать современные web-приложения.

Для написания кода на JavaScript была выбрана интегрированная среда разработки (IDE) Visual Studio Code. Это бесплатный, лёгкий и расширяемый редактор кода от Microsoft, предназначенный для разработки программного обеспечения. Он поддерживает множество языков программирования, имеет встроенную поддержку отладки, подсветку синтаксиса, автодополнение кода и интеграцию с системами контроля версий [21]. Благодаря расширениям Visual Studio Code можно настроить под любые задачи и технологии.

6 этап – написание программного кода.

Структура виджета:

* каталог js - содержит JavaScript-логику работы виджета – всё, что связано с поведением, обработкой данных и взаимодействием с пользователем; именно здесь находятся основные модули, реализующие функциональность редактирования, работу с событиями и состоянием виджета:
  + модуль DicDBEdit\_control.js – содержит основной JavaScript-код для управления поведением виджета: обработчики событий, работа с DOM, логика редактирования данных;
  + модуль DicDBEdit\_singleton.js – реализует синглтон – глобальный объект или класс, который управляет состоянием и настройками виджета, чтобы обеспечить единый доступ к общей информации или методам в разных частях приложения;
* каталог css – хранит стили оформления виджета – отвечает за внешний вид всех элементов: таблиц, форм, кнопок и прочих частей интерфейса; здесь размещаются файлы с CSS, которые подключаются вместе с виджетом:
  + модуль DicDBEdit.css – описывает внешний вид виджета: стили для элементов интерфейса, включая таблицы, кнопки, поля ввода и т. д., позволяя сделать отображение данных удобным и единообразным;
* каталог language – содержит языковые файлы для локализации интерфейса; каждый файл в этом каталоге – это JSON с переводами для одного языка, позволяет добавлять новые языки, не изменяя основной код виджета;
  + модуль ru.json – файл с переводом текстов интерфейса на русский язык, используется для мультиязычности – для подписей кнопок, сообщений об ошибках, подсказок и т. д.;
* модуль $bootstrap.json – конфигурационный файл, который содержит настройки «загрузки» виджета: какие модули подключать, порядок инициализации, пути к ресурсам, параметры по умолчанию; используется для централизованного управления запуском и настройками виджета.

В качестве индивидуального вклада в проект для реализации базовой логики и инициализации виджета были разработаны следующие методы:

* методы инициализации виджета:
  + onBuild() – основная точка сборки виджета: создаёт структуру DOM-элементов, подключает обработчики событий, инициирует загрузку данных, применяет параметры из конфигурации (рис. 10);

ns.DicDBEdit.Singleton.onBuild = function(control) {

this.initParams(control);

control.DOMStyles.width = '100%';

control.minWidth = this.getMinWidth(control);

control.minHeight = this.getMinHeight(control);

control.DOMClasses.push('vcm-widget-table');

this.updateMinMax(control);

setTimeout(() => {

this.initializeEvents(control);

this.initHistory(control);

if (this.getShowHistory(control)) {

const $el = this.getJQElement(control);

const html = this.generateHistoryItems(control);

$el.find('.history-table tbody').html(html);

}

}, 100);};

Рис. 10. Функция onBuild()

* + initParams() – устанавливает начальные параметры виджета из переданных настроек или значений по умолчанию, готовит внутренние переменные для дальнейшей работы (рис. 11);

ns.DicDBEdit.Singleton.initParams = function(control) {

if (!control.param\_bindings) {

control.param\_bindings = {};

}

const registeredParams = ['PrimaryColor', 'ShowHistory', 'ShowCreateForm', 'UniqueField'];

registeredParams.forEach(param => {

if (!control.param\_bindings[param]) {

console.error(`Parameter ${param} is not registered in widget metadata`);

}

});

Рис. 11. Функция initParams()

* методы реализации истории запросов:
  + initHistory() – инициализирует модуль истории: создаёт структуру данных для хранения изменений, настраивает лимиты истории (рис. 12);

ns.DicDBEdit.Singleton.initHistory = function(control) {

if (!control.history) {

control.history = [];

}

};

Рис. 12. Функция initHistory()

* + addToHistory() – добавляет новый запрос пользователя в историю, используется для отображения истории запросов (рис. 13);

ns.DicDBEdit.Singleton.addToHistory = function(control, obj) {

this.initHistory(control);

if (control.history.length === 0 || control.history[0].id !== obj.id) {

control.history.unshift(obj);

if (control.history.length > 10) {

control.history.length = 10;

}

// === Динамическое обновление DOM истории ===

const $el = this.getJQElement(control);

const $historyContainer = $el.find('.history-container');

if ($historyContainer.is(':visible')) {

const html = control.history.map(item =>

`<tr><td>${item.name} // ${item.description} // ${item.category}</td></tr>`

).join('');

$historyContainer.find('tbody').html(html);

}

}

};

Рис. 13. Функция addToHistory()

* + generateHistoryItems() – генерирует и возвращает список элементов истории для отображения пользователю в виде списка запросов (рис. 14);

ns.DicDBEdit.Singleton.generateHistoryItems = function(control) {

const items = control.param\_bindings.HistoryItems?.value || [

'Пример часто используемой записи 1',

'Пример часто используемой записи 2'

];

return items.map(item => `<tr><td class="history-cell">${item}</td></tr>`).join('');

};

Рис. 14. Функция generateHistoryItems()

* + getShowHistory() – геттер, возвращает текущее состояние флага «показывать ли историю» в интерфейсе (рис. 15);

ns.DicDBEdit.Singleton.getShowHistory = function(control) {

return control.param\_bindings.ShowHistory?.value ?? true;

};

Рис. 15. Функция getShowHistory()

* + setShowHistory() – сеттер, устанавливает состояние флага «показывать историю» - включает или отключает отображение истории действий (рис. 16);

ns.DicDBEdit.Singleton.setShowHistory = function(control, value) {

control.param\_bindings.ShowHistory.value = value;

if (!control.id) return;

const $el = this.getJQElement(control);

$el.find('.main-content').css('grid-template-columns', value ? '1fr 300px' : '1fr 0');

$el.find('.history-container').toggle(value);

};

Рис. 16. Функция setShowHistory()

* геттеры и сеттеры параметров:
  + getPrimaryColor() – геттер для основного цвета оформления виджета (для подсветки активных элементов, кнопок и т. д.) (рис. 17);

ns.DicDBEdit.Singleton.getPrimaryColor = function(control) {

return control.param\_bindings.PrimaryColor?.value || '#3E4FA7';

};

Рис. 17. Функция getPrimaryColor()

* + setPrimaryColor() – сеттер для основного цвета оформления виджета (для подсветки активных элементов, кнопок и т. д.) (рис. 18);

ns.DicDBEdit.Singleton.setPrimaryColor = function(control, value) {

control.param\_bindings.PrimaryColor.value = value;

if (!control.id) return;

const $el = this.getJQElement(control);

Рис. 18. Функция setPrimaryColor()

Рис. 18. Продолжение

$el.find(`

.search-container,

.table-container,

.create-record-container,

.history-container,

.search-input,

.attribute-input,

.filter-input,

.create-record-input

`).css('border-color', value);

$el.find(`

.search-button,

.btn,

.create-record-button

`).css('background-color', value);

$el.find('.first').css('border-bottom-color', value);

};

* + getMinHeight() – геттер для минимальной высоты виджета – возвращает параметр, задающий ограничения по высоте интерфейса (рис. 19);

ns.DicDBEdit.Singleton.getMinHeight = () => 600;

Рис. 19. Функция getMinHeight()

* + getMinWidth() – геттер для минимальной ширины виджета – возвращает параметр, задающий ограничения по ширине интерфейса (рис. 20);

ns.DicDBEdit.Singleton.getMinWidth = () => 1000;

Рис. 20. Функция getMinWidth()

* + setShowCreateForm() – сеттер для флага «показывать ли форму создания записи», управляет видимостью формы добавления данных в словарь (рис. 21);

ns.DicDBEdit.Singleton.setShowCreateForm = function(control, value) {

control.param\_bindings.ShowCreateForm.value = value;

if (!control.id) return;

this.getJQElement(control).find('.create-record-container').toggle(value);

};

Рис. 21. Функция setShowCreateForm()

* моки («заглушки» для тестирования) [10]:
  + mockGridSQL() – мок-метод, эмулирующий запрос данных таблицы из базы данных, возвращает фиктивные данные для тестирования интерфейса без подключения к настоящей базе данных (рис. 22);

function mockGridSQL() {

return new Promise((resolve) => {

setTimeout(() => resolve(mockGridDictionary), 200);

});

}

Рис. 22. Функция mockGridSQL()

* + mockSearchSQL() – мок-метод для эмуляции поиска по словарю через SQL-запрос, используется для отработки сценариев поиска в интерфейсе (рис. 23);

function mockSearchSQL(query) {

return new Promise((resolve) => {

const results = mockGridDictionary.filter(item =>

item.name.toLowerCase().includes(query.toLowerCase())

);

resolve(results);

});

}

Рис. 23. Функция mockSearchSQL()

* + mockGridDictionary – массив, предоставляющий фиктивный словарь данных для отображения в гриде, удобен для разработки интерфейса до подключения реального API (рис. 24);

const mockGridDictionary = [

{ id: 1, name: 'Кот', description: 'Млекопитающее', category: 'Животное' },

{ id: 2, name: 'Собака', description: 'Друг человека', category: 'Животное' },

{ id: 3, name: 'Кошка', description: 'Ленивое существо', category: 'Животное' },

{ id: 4, name: 'Трактор', description: 'Машина для обработки земли', category: 'Техника' }

];

Рис. 24. Массив mockGridDictionary

* базовые вспомогательные методы:
  + getJQElement() – возвращает jQuery-элемент по заданному селектору или DOM-узлу, служит для унифицированного доступа к элементам интерфейса в коде (рис. 25).

ns.DicDBEdit.Singleton.getJQElement = function(control) {

return $(`#${control.Name}`);

};

Рис. 25. Функция getJQElement()

Интерфейс виджета представлен на рис. 26.

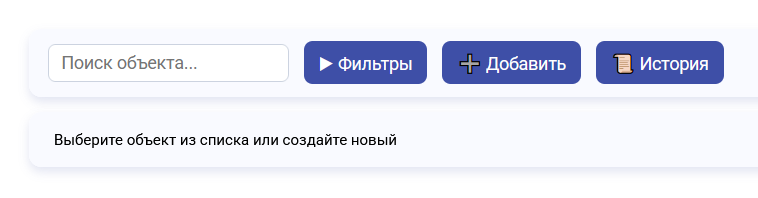


Рис. 26. Интерфейс виджета

Полный текст программы представлен в приложении.

7 этап – тестирование и отладка.

На этом этапе проводилась проверка корректности работы всех функций и интерфейса виджета.

В результате тестирования были выявлены и исправлены мелкие ошибки в настройках параметров.

# Раздел 5. Мероприятия по охране труда и технике безопасности

В производственных процессах, связанных с использованием компьютерного оборудования и программного обеспечения, большое значение имеет обеспечение безопасных условий труда и создание благоприятного микроклимата в помещениях.

Микроклимат рабочих помещений должен соответствовать санитарно-гигиеническим требованиям СП 2.2.3670-20 [3]:

* температура воздуха – 22-24 °C;
* относительная влажность – 40-60 %;
* скорость движения воздуха – не более 0,1 м/с.

Для этого в помещениях должны быть предусмотрены системы вентиляции и кондиционирования, обеспечивающие подачу свежего воздуха и поддержание комфортных условий для сотрудников.

К основным опасным и вредным производственным факторам на рабочем месте специалиста, работающего за компьютером, согласно ГОСТ 12.0.003-2015, относятся [1]:

* воздействие электромагнитного излучения от мониторов и оборудования;
* зрительное напряжение при длительной работе за экраном;
* статическое напряжение мышц при продолжительном сидении;
* психоэмоциональное напряжение при высокой интенсивности труда и сжатых сроках выполнения задач;
* риск поражения электрическим током при эксплуатации компьютерного и сетевого оборудования.

Для обеспечения безопасности производственного процесса в соответствии с Приказом Минтруда РФ от 29.10.2021 №771н принимаются следующие мероприятия [2]:

* организация правильного освещения рабочих мест с уровнем освещенности не менее 300–500 лк, чтобы снизить нагрузку на органы зрения;
* использование эргономичной офисной мебели с возможностью регулировки высоты стула и наклона спинки, что способствует правильной осанке и снижает утомляемость;
* проведение регулярных инструктажей по охране труда и технике безопасности с регистрацией в соответствующем журнале;
* введение режима труда и отдыха с перерывами для выполнения гимнастики для глаз и разминки через каждый час работы за компьютером;
* оснащение помещений автоматической пожарной сигнализацией, первичными средствами пожаротушения (огнетушители) и наличием свободных эвакуационных выходов;
* применение устройств защитного отключения (УЗО) и исправной электропроводки для предотвращения поражения электрическим током;
* обеспечение исправности и сертификации используемого оборудования, регулярное техническое обслуживание электросетей и систем электроснабжения.

Соблюдение этих мероприятий позволяет поддерживать безопасность производственного процесса, снижает риск возникновения травм и профессиональных заболеваний, способствует сохранению здоровья сотрудников и созданию комфортной рабочей среды.

# Заключение

В результате прохождения производственной практики были решены следующие задачи:

* пройден инструктаж по охране труда и технике безопасности;
* получены сведения об организации производства в ООО «Совкомбанк Технологии», а также о структуре и функциях подразделения ОРЗКиС;
* выполнено индивидуальное задание: разработан виджет для ФИС-платформы;
* оформлен отчет о прохождении практики.

В результате прохождения практики были освоены следующие компетенции:

* способность использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности (ОПК-2);
* способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно коммуникационных технологий и с учетом основных требований информационной безопасности (ОПК-3).

Предположительная тема выпускной квалификационной работы: «Создание новых виджетов для ФИС платформы».

# Список литературы

1. ГОСТ 12.0.003-2015. Система стандартов безопасности труда. Опасные и вредные производственные факторы. Классификация : нац. Стандарт Рос. Федерации : изд. офиц. : утв. и введ. в действие Приказом Федерального агентства по техническому регулированию и метрологии от 09.06.2016 № 602-ст : введ. впервые : дата введ. 2017-03-01 / разраб. Обществом с ограниченной ответственностью «Экожилсервис», Федер. гос. бюдж. образоват. учрежд. высш. проф. образов. «Пермский национальный исследовательский политехнический университет» — М.: Стандартифнорм, 2017.
2. Об утверждении Примерного перечня ежегодно реализуемых работодателем мероприятий по улучшению условий и охраны труда, ликвидации или снижению уровней профессиональных рисков либо недопущению повышения их уровней : Приказ Министерства труда и социальной защиты РФ от 29.10.2021 №771н.
3. СП 2.2.3670-20. Свод правил. Санитарно-эпидемиологические требования к условиям труда : утв. Постановлением Главного государственного санитарного врача РФ от 02.12.2020 № 40 : введ. 01.01.2021 — М.: Роспотребнадзор, 2020.
4. Андирякова О. О. Применение low-code технологии для решения бизнес-задач / О. О. Андирякова, А. А. Крюкова, М. И. Иваев – Текст : электронный // Индустриальная экономика. – 2023. – № 2.
5. Е.В. Ершов, Л.Н. Виноградова, В.В. Селивановских, О.Л. Селяничев Методика и организация самостоятельной работы студентов: Учеб. пособие. – ФГБОУ ВПО «Череповецкий государственный университет», 2012 г.
6. Флэнаган, Д. JavaScript. Подробное руководство, 5-е изд. : Пер. с англ. – СПб. : Символ-Плюс, 2008. – 992 с.
7. Единая экосистема FIS — Fisgroup [электр.ресурс]. https://fisgroup.ru/blog/ Дата обращения: 23.06.2025.
8. Карьера в Совкомбанк Технологиях – Совкомбанк Технологии [электр.ресурс] https://sovcombank.it/#career. Дата обращения: 18.06.2025.
9. Компания «Совкомбанк Технологии» — Хабр Карьера [электр.ресурс] https://career.habr.com/companies/scbt. Дата обращения: 18.06.2025.
10. Моки, стабы и фейки: в чем разница и когда что использовать? — Хабр [электр.ресурс] https://habr.com/ru/articles/899788/. Дата обращения: 30.06.2025.
11. МТС Линк [электр.ресурс] https://mts-link.ru/ Дата обращения: 18.06.2025.
12. О компании — Совкомбанк Технологии [электр.ресурс] https://sovcombank.it/#about. Дата обращения: 18.06.2025.
13. Портал документации Platform — Case Platform [электр.ресурс] https://doc.caseplatform.tech/. Дата обращения: 24.06.2025.
14. Совкомбанк Технологии — Совком People [электр.ресурс] https://people.sovcombank.ru/it. Дата обращения: 18.06.2025.
15. Технологический стек — Совкомбанк Технологии [электр.ресурс] https://sovcombank.it/#technology. Дата обращения: 18.06.2025.
16. Учимся работать с платформой — Rutube [электр.ресурс] https://rutube.ru/plst/393933/. Дата обращения: 24.06.2025.
17. Что такое DevOps? — Microsoft Learn [электр.ресурс] https://learn.microsoft.com/ru-ru/devops/what-is-devops. Дата обращения: 20.06.2025.
18. Agile: что это такое и где используется, принципы методологии — Яндекс.Практикум [электр.ресурс] https://practicum.yandex.ru/blog/metodology-agile/. Дата обращения: 20.06.2025.
19. FIS Platform: low-code платформа для автоматизации финансового сектора от компании FIS — IaaSSaaSPaaS [электр.ресурс] https://iaassaaspaas.ru/rating/low-code-sistemy/fis-platform-low-code-platforma-dlya-avtomatizatsii-finansovogo-sektora-ot-kompanii-fis. Дата обращения: 23.06.2025.
20. Redmine [электр.ресурс] https://redmine.scb-lab.ru/. Дата обращения: 17.06.2025.
21. Tutorial: Get started with Visual Studio Code — Visual Studio Code [электр.ресурс] https://code.visualstudio.com/docs/getstarted/getting-started. Дата обращения: 26.06.2025.

# Приложение

Текст программы

Текст модуля DicDBEdit\_singleton.js представлен на рис. П1.1.

js.namespace('JQ.Widget.DicDBEdit');

js.namespace('JQ.Widget');

ns.DicDBEdit.Singleton = js.clone(ns.Base.Singleton);

// ====================== Инициализация ======================

ns.DicDBEdit.Singleton.onBuild = function(control) {

this.initParams(control);

control.DOMStyles.width = '100%';

control.minWidth = this.getMinWidth(control);

control.minHeight = this.getMinHeight(control);

control.DOMClasses.push('vcm-widget-table');

this.updateMinMax(control);

setTimeout(() => {

this.initializeEvents(control);

this.initHistory(control);

if (this.getShowHistory(control)) {

const $el = this.getJQElement(control);

const html = this.generateHistoryItems(control);

$el.find('.history-table tbody').html(html);

}

}, 100);};

ns.DicDBEdit.Singleton.initParams = function(control) {

if (!control.param\_bindings) {

control.param\_bindings = {};

}

const registeredParams = ['PrimaryColor', 'ShowHistory', 'ShowCreateForm', 'UniqueField'];

registeredParams.forEach(param => {

if (!control.param\_bindings[param]) {

console.error(`Parameter ${param} is not registered in widget metadata`);

}

});

const params = [

{ qname: 'primaryColor', default\_value: '#3E4FA7' },

{ qname: 'showHistory', default\_value: true },

{ qname: 'showCreateForm', default\_value: true },

{ qname: 'uniqueField', default\_value: 'name' }

];

params.forEach(param => {

if (!(param.qname in control.param\_bindings)) {

control.param\_bindings[param.qname] = { value: param.default\_value };

}

});

};

ns.DicDBEdit.Singleton.initHistory = function(control) {

if (!control.history) {

control.history = [];

Рис. П1.1. Текст модуля DicDBEdit\_singleton.js

Рис. П1.1. Продолжение

}

};

ns.DicDBEdit.Singleton.addToHistory = function(control, obj) {

this.initHistory(control);

if (control.history.length === 0 || control.history[0].id !== obj.id) {

control.history.unshift(obj);

if (control.history.length > 10) {

control.history.length = 10;

}

// === Динамическое обновление DOM истории ===

const $el = this.getJQElement(control);

const $historyContainer = $el.find('.history-container');

if ($historyContainer.is(':visible')) {

const html = control.history.map(item =>

`<tr><td>${item.name} // ${item.description} // ${item.category}</td></tr>`

).join('');

$historyContainer.find('tbody').html(html);

}

}

};

function mockSearchSQL(query) {

return new Promise((resolve) => {

const results = mockGridDictionary.filter(item =>

item.name.toLowerCase().includes(query.toLowerCase())

);

resolve(results);

});

}

// Мок-данные для gridSQL

const mockGridDictionary = [

{ id: 1, name: 'Кот', description: 'Млекопитающее', category: 'Животное' },

{ id: 2, name: 'Собака', description: 'Друг человека', category: 'Животное' },

{ id: 3, name: 'Кошка', description: 'Ленивое существо', category: 'Животное' },

{ id: 4, name: 'Трактор', description: 'Машина для обработки земли', category: 'Техника' }

];

function mockGridSQL() {

return new Promise((resolve) => {

setTimeout(() => resolve(mockGridDictionary), 200);

});

}

ns.DicDBEdit.Singleton.getMarkupContent = function(control) {

return `

<div class="dic-db-edit" id="${control.Name}">

<div class="main-toolbar">

<input type="text" class="main-search-input" placeholder="Поиск объекта..." />

<button class="btn open-filter-popup">▶ Фильтры</button>

<button class="btn open-create-popup">➕ Добавить</button>

<button class="btn toggle-history">📜 История</button>

</div>

<div class="info-label">Выберите объект из списка или создайте новый</div>

<div class="main-table-container">

<table class="main-table">

Рис. П1.1. Продолжение

<thead>

<tr>

<th></th>

<th></th>

</tr>

</thead>

<tbody></tbody>

</table>

</div>

<div class="history-container" style="display: none;">

<table class="history-table">

<thead><tr><th>История</th></tr></thead>

<tbody></tbody>

</table>

</div>

<div class="popup-container" id="popup-container"></div>

</div>

`;

};

// ====================== Popup шаблоны ======================

ns.DicDBEdit.Singleton.getFilterPopupHtml = function(control) {

return `

<div class="filter-table-popup popup-box">

<div class="popup-header">

<span>Фильтр</span>

<button class="btn-close">✖</button>

</div>

<div class="popup-body">

<label>Поиск: <input type="text" class="filter-input" /></label>

<div class="popup-buttons">

<button class="btn select-record">Выбрать</button>

<button class="btn add-record">Добавить</button>

</div>

</div>

</div>

`;

};

ns.DicDBEdit.Singleton.getCreatePopupHtml = function(control) {

return `

<div class="create-record-popup popup-box">

<div class="popup-header">

<span>Создать запись</span>

<button class="btn-close-create">✖</button>

</div>

<div class="popup-body">

<label>Название: <input type="text" class="create-name" /></label>

<label>Описание: <input type="text" class="create-desc" /></label>

<label>Категория: <input type="text" class="create-category" /></label>

<button class="btn-save-record">Сохранить</button>

</div>

</div>

`;

};

// ====================== Обработчики событий ======================

ns.DicDBEdit.Singleton.initializeEvents = function(control) {

Рис. П1.1. Продолжение

const $el = this.getJQElement(control);

const $popupContainer = $el.find('#popup-container');

const $searchInput = $el.find('.main-search-input');

$searchInput.on('blur', () => {

const searchValue = $searchInput.val().trim();

if (!searchValue) return;

mockSearchSQL(searchValue).then(results => {

if (results.length === 1) {

const result = results[0];

console.log('Одна запись найдена:', result);

$el.find('.info-label').text(`${result.id} // ${result.name} // ${result.description} // ${result.category}`);

this.addToHistory(control, result);

} else if (results.length > 1) {

console.log('Несколько записей:', results);

$el.find('.info-label').text(`🔍 Найдено несколько совпадений (${results.length})`);

this.renderFilterTable(control, results);

} else {

$el.find('.info-label').text(`❌ Ничего не найдено`);

}

});

});

$el.find('.open-filter-popup').on('click', () => {

this.GridSQL().then(data => {

this.renderFilterTable(control, data);

});

});

$el.find('.open-create-popup').on('click', () => {

this.GridSQL().then(data => {

this.openCreateRecord(control, data, $popupContainer);

});

});

// Кнопка "История"

$el.find('.toggle-history').on('click', () => {

const $history = $el.find('.history-container');

if ($history.is(':hidden')) {

this.initHistory(control);

const html = control.history.map(item => `<tr><td>${item.name} // ${item.description} // ${item.category}</td></tr>`).join('');

$history.find('tbody').html(html);

}

$history.toggle();

$el.find('.history-container').on('click', '.history-table tbody tr', (event) => {

const $row = $(event.currentTarget);

const index = $row.index();

const obj = control.history && control.history[index];

if (!obj) return;

$el.find('.main-search-input').val(obj.name);

$el.find('.info-label').text(`${obj.id} // ${obj.name} // ${obj.description} // ${obj.category}`);

control.selected\_value = obj;

Рис. П1.1. Продолжение

});

});

};

ns.DicDBEdit.Singleton.openCreateRecord = function(control, data, $popupContainer) {

let $popup = $popupContainer.find('.create-record-popup');

if (!$popup.length) {

$popup = $(this.getCreatePopupHtml(control));

$popupContainer.append($popup);

}

$popup.fadeIn(200);

$popup.find('.btn-close-create').off('click').on('click', () => $popup.fadeOut(200));

$popup.find('.btn-save-record').off('click').on('click', () => {

const name = $popup.find('.create-name').val().trim();

const desc = $popup.find('.create-desc').val().trim();

const category = $popup.find('.create-category').val().trim();

if (!name) {

alert('Название обязательно');

return;

}

const newId = data.length ? Math.max(...data.map(d => d.id)) + 1 : 1;

const newObj = {

id: newId,

name: name,

description: desc,

category: category

};

data.push(newObj);

this.renderFilterTable(control, data);

this.addToHistory(control, newObj);

$popup.fadeOut(200);

});

};

ns.DicDBEdit.Singleton.renderFilterTable = function(control, data) {

const $el = this.getJQElement(control);

const $popupContainer = $el.find('#popup-container');

$popupContainer.find('.filter-table-popup').remove();

const keys = data.length > 0 ? Object.keys(data[0]) : [];

const headersHtml = keys.map(key => `<th>${key}</th>`).join('');

const filtersHtml = keys.map(key => `<th><input type="text" class="filter-input" data-field="${key}" placeholder="Фильтр"></th>`).join('');

const rowsHtml = data.map(item => {

const cells = keys.map(key => `<td>${item[key]}</td>`).join('');

return `<tr data-id="${item.id}">${cells}</tr>`;

}).join('');

const popupHtml = `

<div class="filter-table-popup popup-box" style="display:none;">

<div class="popup-header">

<span>Фильтр</span>

<button class="btn-close">✖</button>

</div>

<div class="popup-body">

Рис. П1.1. Продолжение

<table class="main-table">

<thead>

<tr>${headersHtml}</tr>

<tr class="filter-row">${filtersHtml}</tr>

</thead>

<tbody>${rowsHtml}</tbody>

</table>

<div class="popup-buttons">

<button class="btn select-record" disabled>Выбрать</button>

<button class="btn add-record">Добавить</button>

</div>

</div>

</div>

`;

const $popup = $(popupHtml);

$popupContainer.append($popup);

$popup.fadeIn(200);

$popup.find('thead .filter-input').on('input', function() {

const filters = {};

$popup.find('thead .filter-input').each(function() {

const field = $(this).data('field');

const val = $(this).val().toLowerCase();

if(val) filters[field] = val;

});

$popup.find('tbody tr').each(function() {

const $tr = $(this);

let visible = true;

for (const [field, val] of Object.entries(filters)) {

const idx = keys.indexOf(field);

const cellText = $tr.find('td').eq(idx).text().toLowerCase();

if(!cellText.includes(val)) {

visible = false;

break;

}

}

$tr.toggle(visible);

});

});

$popup.find('.btn-close').on('click', () => $popup.fadeOut(200));

$popup.find('.select-record').on('click', () => {

const $selectedRow = $popup.find('tbody tr.selected');

if ($selectedRow.length === 0) return;

const selectedId = $selectedRow.data('id');

const selectedObj = data.find(item => item.id === selectedId);

if (!selectedObj) return;

const $searchInput = $el.find('.main-search-input');

$searchInput.val(selectedObj[control.param\_bindings.UniqueField?.value || 'name']);

const labelText = Object.values(selectedObj).join(' // ');

$el.find('.info-label').text(labelText);

control.selected\_value = selectedObj;

Рис. П1.1. Продолжение

this.addToHistory(control, selectedObj);

$popup.fadeOut(200);

});

$popup.find('.add-record').on('click', () => {

console.log('Добавить новую запись...');

$popup.fadeOut(200);

this.openCreateRecord(control, data, $popupContainer);

});

$popup.find('tbody tr').on('click', function() {

$popup.find('tbody tr').removeClass('selected');

$(this).addClass('selected');

$popup.find('.select-record').prop('disabled', false);

});

};

// ====================== Методы генерации DOM ======================

ns.DicDBEdit.Singleton.generateTableHeaders = function(control) {

const columns = control.param\_bindings.Columns?.value || ['column1', 'column2', 'column3', 'column4'];

return columns.map(col => `<th class="col">${col}</th>`).join('');

};

ns.DicDBEdit.Singleton.generateTableFilters = function(control, color) {

const count = control.param\_bindings.Columns?.value?.length || 4;

return `

<tr class="inner">

${Array(count).fill().map(() => `

<th class="filter">

<input type="text" class="filter-input" placeholder="filtertext" style="border-color: ${color}">

</th>`).join('')}

</tr>

`;

};

ns.DicDBEdit.Singleton.generateSampleData = function(control) {

const rows = control.param\_bindings.SampleData?.value || [['some', 'something', 'more something', 'maybe date type']];

const rowHtml = rows.map(row => `

<tr class="inner">${row.map(cell => `<td>${cell}</td>`).join('')}</tr>

`).join('');

return rowHtml + `

<tr class="button\_line last">

<td colspan="${rows[0]?.length || 4}">

<button class="btn" style="background-color: ${this.getPrimaryColor(control)}">▶</button>

<button class="btn" style="background-color: ${this.getPrimaryColor(control)}">➕</button>

</td>

</tr>

`;

};

ns.DicDBEdit.Singleton.generateCreateForm = function(control, color) {

const fields = control.param\_bindings.CreateFormFields?.value || 3;

const inputs = Array(fields).fill().map((\_, i) => `

<input class="create-record-input" type="text" placeholder="Значение атрибута${i+1}" style="border-color: ${color}">`).join('');

return inputs + `<button class="create-record-button" style="background-color: ${color}">Создать</button>`;

};

Рис. П1.1. Продолжение

ns.DicDBEdit.Singleton.generateHistoryItems = function(control) {

const items = control.param\_bindings.HistoryItems?.value || [

'Пример часто используемой записи 1',

'Пример часто используемой записи 2'

];

return items.map(item => `<tr><td class="history-cell">${item}</td></tr>`).join('');

};

// ====================== Работа с DOM ======================

ns.DicDBEdit.Singleton.getJQElement = function(control) {

return $(`#${control.Name}`);

};

// ====================== Геттеры и сеттеры ======================

ns.DicDBEdit.Singleton.getPrimaryColor = function(control) {

return control.param\_bindings.PrimaryColor?.value || '#3E4FA7';

};

ns.DicDBEdit.Singleton.getShowHistory = function(control) {

return control.param\_bindings.ShowHistory?.value ?? true;

};

ns.DicDBEdit.Singleton.getShowCreateForm = function(control) {

return control.param\_bindings.ShowCreateForm?.value ?? true;

};

ns.DicDBEdit.Singleton.getMinHeight = () => 600;

ns.DicDBEdit.Singleton.getMinWidth = () => 1000;

ns.DicDBEdit.Singleton.setPrimaryColor = function(control, value) {

control.param\_bindings.PrimaryColor.value = value;

if (!control.id) return;

const $el = this.getJQElement(control);

$el.find(`

.search-container,

.table-container,

.create-record-container,

.history-container,

.search-input,

.attribute-input,

.filter-input,

.create-record-input

`).css('border-color', value);

$el.find(`

.search-button,

.btn,

.create-record-button

`).css('background-color', value);

$el.find('.first').css('border-bottom-color', value);

};

ns.DicDBEdit.Singleton.setShowHistory = function(control, value) {

control.param\_bindings.ShowHistory.value = value;

if (!control.id) return;

Рис. П1.1. Продолжение

const $el = this.getJQElement(control);

$el.find('.main-content').css('grid-template-columns', value ? '1fr 300px' : '1fr 0');

$el.find('.history-container').toggle(value);

};

ns.DicDBEdit.Singleton.setShowCreateForm = function(control, value) {

control.param\_bindings.ShowCreateForm.value = value;

if (!control.id) return;

this.getJQElement(control).find('.create-record-container').toggle(value);

};

// ====================== Таблица и История ======================

ns.DicDBEdit.Singleton.addTableRow = function(control, rowData) {

if (!control.id) return;

const $tbody = this.getJQElement(control).find('.table-container tbody');

const $buttonRow = $tbody.find('.button\_line').detach();

const rowHtml = `<tr class="inner">${rowData.map(cell => `<td>${cell}</td>`).join('')}</tr>`;

$tbody.append(rowHtml).append($buttonRow);

};

ns.DicDBEdit.Singleton.clearTable = function(control) {

if (!control.id) return;

const $tbody = this.getJQElement(control).find('.table-container tbody');

const $buttonRow = $tbody.find('.button\_line').detach();

$tbody.empty().append($buttonRow);

};

ns.DicDBEdit.Singleton.addHistoryItem = function(control, item) {

if (!control.id) return;

const $tbody = this.getJQElement(control).find('.history-table tbody');

$tbody.append(`<tr><td class="history-cell">${item}</td></tr>`);

};

Текст модуля DicDBEdit\_control.js представлен на рис. П1.2.

/\*\*

\* В данном файле находится описание пользовательского интерфейса

\*

\* Описание свойств header.params:

\* qname[String] - системное имя свойства

\*

\* description[string] - описание свойства, оно отображается ввиде тултипа при наведение

\* мышки на данную настройку в арме конфигураций

\*

\* type[setting/field] - тип свойства обозначающий позволяет ли данное свойство сохранить значение на сервер

\* (setting - обычное свойство, field - свойство через которое можно сохранить значение на сервер)

\*

\* editor\_type[string|number|boolean|etc] - тип редактора через которое задаётся свойство в арме конфигураций

Рис. П1.2. Текст модуля DicDBEdit\_control.js

Рис. П1.2. Продолжение

\*

\* default\_value[all] - базовое значение свойства, поддерживает любой тип js примитивов

\*

\* Остальные свойства служат для идентификации виджета в системе:

\* control\_id - идентификатора контрола(виджета)

\* widget.script - идентификатора js скрита, нужен для идентификации виджета в редакторах арме конфигураций

\* widget.name - имя класс который вызывается при создании экземпляра виджета

\*/

js.namespace('JQ.Widget.DicDBEdit');

ns.Control = {

control\_id: 'DicDBEdit.control',

header:

{

params: [// Основные параметры таблицы

{

qname: 'PrimaryColor',

qnameRU: 'Основной цвет',

description: 'Основной цвет интерфейса',

type: 'setting',

editor\_type: 'textColorpicker',

default\_value: '#3E4FA7',

},

{

qname: 'ShowHistory',

qnameRU: 'Показывать историю',

description: 'Показывать панель с историей выбора',

type: 'setting',

editor\_type: 'boolean',

default\_value: 'true'

},

{

qname: 'ShowCreateForm',

qnameRU: 'Показывать форму создания',

description: 'Показывать форму создания новой записи',

type: 'setting',

editor\_type: 'boolean',

default\_value: 'true'

},

{

qname: 'UniqueField',

qnameRU: 'Имя уникального поля для записи',

description: 'Имя уникального поля для записи',

type: 'setting',

editor\_type: 'string',

default\_value: 'name'

},{

qname: 'TableConfig',

qnameRU: 'Конфигурация таблицы',

description: 'Настройка структуры таблицы, фильтра и количества полей. Пример: {"columns":[{"name":"id","type":"number"},{"name":"name","type":"string"}],"filter":{"status":"active"},"visibleFieldCount":2}',

type: 'setting',

editor\_type: 'text',

default\_value: '{"columns":[{"name":"id","type":"number"},{"name":"name","type":"string"}],"filter":{"status":"active"},"visibleFieldCount":2}'

}],

events: [],

templates: []

Рис. П1.2. Продолжение

},

type: "DicDBEdit",

widget:

{

script: 'DicDBEdit.widget',

name: 'DicDBEdit'

}

};

Текст модуля DicDBEdit.css представлен на рис. П1.3.

|  |  |
| --- | --- |
| body {  font-family: 'Segoe UI', sans-serif;  margin: 0;  padding: 20px;  display: grid;  /\*grid-template-columns: 1fr 300px;\*/  grid-gap: 20px;  max-width: 1400px;  margin: 0 auto;  background: #f4f7fb;  }  .main-toolbar {  display: flex;  gap: 12px;  align-items: center;  margin-bottom: 12px;  padding: 10px 15px;  background: #f9faff;  border-radius: 10px;  box-shadow: 0 4px 8px rgba(62, 79, 167, 0.15);  transition: box-shadow 0.3s ease;  }  .info-label:hover,  .search-container:hover,  .create-record-container:hover,  .history-container:hover,  .filter-table-popup:hover {  box-shadow: 0 6px 12px rgba(62, 79, 167, 0.2);  }  .info-label,  .search-container,  .create-record-container,  .history-container,  .filter-table-popup {  background: #f9faff;  border-radius: 10px;  padding: 15px 20px;  box-shadow: 0 4px 8px rgba(62, 79, 167, 0.15);  transition: box-shadow 0.3s ease;  } | .popup-buttons {  margin-top: 12px;  display: flex;  justify-content: flex-end;  gap: 8px;  }  .main-content {  display: flex;  flex-direction: column;  gap: 20px;  }  .table-container {  border: solid 1px #3E4FA7;  border-radius: 7px;  width: 100%;  overflow: hidden;  }  .table-title {  color: gray;  font-size: 14px;  margin-bottom: 5px;  }  table {  width: 100%;  border-collapse: collapse;  }  th, td {  border: none;  padding: 10px;  line-height: 15px;  text-align: left;  }  .inner {  border-bottom: 5px solid #f0f0f0;  height: 35px;  } |

Рис. П1.3. Текст модуля DicDBEdit.css

Рис. П1.3. Продолжение

|  |  |
| --- | --- |
| /\*.penultimate {  height: 35px;  } \*/  .first {  border-bottom: solid 1px #3E4FA7;  }  .last {  border-top: solid 1px #3E4FA7;  }  .col {  text-align: center;  }  .filter {  background: #f0f0f0;  }  .filter-input {  width: 100%;  padding: 4px;  box-sizing: border-box;  border: 1px solid #ddd;  border-radius: 4px;  }  .btn {  background: #3E4FA7;  color: white;  border: none;  padding: 8px 12px;  border-radius: 7px;  font-size: 14px;  cursor: pointer;  }  .btn:hover {  background: #2c3a7a;  }  .button\_line {  background: #f0f0f0;  }  .search-input {  border: solid 1px #3E4FA7;  border-radius: 5px;  width: 100%;  height: 30px;  padding: 3px 10px;  } | .search-button {  background-color: #3E4FA7;  color: white;  border: none;  border-radius: 5px;  cursor: pointer;  width: 30px;  height: 30px;  flex-shrink: 0;  }  .attribute-input {  border: dashed 1px #3E4FA7;  padding: 3px 10px;  border-radius: 5px;  width: 100%;  height: 30px;  }  .create-record-input {  border: solid 1px #3E4FA7;  border-radius: 5px;  width: 100%;  height: 30px;  padding: 3px 10px;  }  .create-record-button {  background-color: #3E4FA7;  color: white;  border: none;  border-radius: 5px;  cursor: pointer;  padding: 8px;  width: 130px;  margin-top: 10px;  }  .history-table {  width: 100%;  border-collapse: collapse;  display: block;  overflow-y: auto;  flex-grow: 1;  }  .history-header {  font-weight: bold;  background-color: #f0f0f0;  position: sticky;  top: 0;  width: 100%;  }  .history-cell {  height: 35px;  border-bottom: 1px solid #f0f0f0;  padding: 8px 12px;  } |

Рис. П1.3. Продолжение

|  |  |
| --- | --- |
| .history-table::-webkit-scrollbar {  width: 8px;  }  .history-table::-webkit-scrollbar-track {  background: #f1f1f1;  border-radius: 4px;  }  .history-table::-webkit-scrollbar-thumb {  background: #3E4FA7;  border-radius: 4px;  }  .history-table::-webkit-scrollbar-thumb:hover {  background: #2c3a7a;  } | input[type="text"], .filter-input, .create-record-input, .attribute-input {  border: 1px solid #ccd3e0;  border-radius: 6px;  padding: 6px 10px;  font-size: 14px;  background: #fff;  }  /\* Заголовки таблиц и фильтров \*/  th {  font-weight: 600;  color: #3E4FA7;  }  /\* Выделение строк \*/  tr.selected {  background-color: #e0e6f9;  } |

Текст модуля ru.json представлен на рис. П1.4.

{

"controls": {

"DicDBEdit": {

"params": {

"TableBorderColor": {

"qname": "ЦветГраницы"

},

"TableHeaderColor": {

"qname": "ЦветЗаголовка"

}

},

"events": {}

}

}

}

Рис. П1.4. Текст модуля ru.json

Текст модуля $bootstrap.json представлен на рис. П1.5.

{

control: {

widgets: [

{

name: 'extra',

label: 'Дополнительно',

children: [

{

control\_id: "DicDBEdit.control",

label: "Dictionary Database Edit",

class\_name: "DicDBEdit"

Рис. П1.5. Текст модуля $bootstrap.json

Рис. П1.5. Продолжение

}

]

}

]

},

js: [

{

file: "DicDBEdit\_singleton.js"

},

{

file: "DicDBEdit\_control.js"

}

],

css: [

{

file: "DicDBEdit.css"

}

],

language: [

{

file: 'ru.json'

}

]

}