Московский государственный университет имени М.В. Ломоносова

Филиал МГУ в г. Дубне

01.04.02 Прикладная математика и информатика

Магкаев Ролен Арсенович

Разработка базы данных состояний и калибровок для эксперимента SPD на коллайдере NICA

КУРСОВАЯ РАБОТА

Научный руководитель:

с.н.с. ЛЯП ОИЯИ, к. ф.-м. н.

Ф.В. Прокошин

Дубна, 2026

Аннотация

В работе описывается прототип базы данных Conditions & Calibrations для эксперимента SPD на коллайдере NICA. Необходимость такой базы определяется требованием централизованно хранить параметры состояния детектора, калибровки и конфигурации, без которых реконструкция событий не воспроизводима. Цель работы — спроектировать и реализовать веб-приложение с разграничением доступа по ролям, управлением проектами, группами участников и табличными данными пользователей. Решение построено на Django, PostgreSQL и развертывается в Docker. В прототипе реализованы создание проектов, назначение групп, управление таблицами с типизированными колонками, вложенные таблицы, экспорт данных и проектная заморозка пользователей. Полученный каркас предполагается использовать для последующей интеграции интервалов валидности, версий и неизменяемых payload-объектов.

Ключевые слова: conditions and calibrations, SPD, NICA, PostgreSQL, разграничение доступа, проектные данные, вложенные таблицы, веб-приложение.

Annotation

The paper describes a prototype of the Conditions & Calibrations database for the SPD experiment at the NICA collider. The need for such a database is determined by the requirement to centrally store detector status, calibration, and configuration parameters, without which event reconstruction is not reproducible. The purpose of the work is to design and implement a web application with role-based access control, project management, participant groups, and tabular user data. The solution is built on Django, PostgreSQL and deployed in Docker. The prototype implements project creation, group assignment, table management with typed columns, nested tables, data export, and project freezing of users. The resulting framework is supposed to be used for the subsequent integration of validity intervals, versions, and immutable payload objects.

Keywords: conditions and calibrations, SPD, NICA, PostgreSQL, access control, project data, nested tables, web application.

Оглавление

Введение

Современные эксперименты физики высоких энергий работают не только с событийными данными, но и со значительным объемом служебной информации, без которой реконструкция физических процессов даст невоспроизводимый результат. Сюда относятся параметры состояния детектора, результаты калибровок, конфигурации программного обеспечения, версии алгоритмов реконструкции и интервалы времени, в течение которых эти значения считаются действительными. То есть ошибка в выборе таких данных может привести к различиям в реконструкции даже на одном и том же наборе зарегистрированных событий.

Эксперимент SPD [1] на коллайдере NICA ориентирован на изучение спиновой структуры нуклона и связанных с ней задач ядерной физики. Для таких установок воспроизводимость обработки данных особенно важна: один и тот же набор событий должен реконструироваться с явно зафиксированным набором условий, калибровок и конфигураций. Поэтому в программной инфраструктуре эксперимента нужна специализированная база данных, которая хранит эти сведения и предоставляет их пользователям и вычислительным сервисам.

База данных Conditions & Calibrations — один из таких компонентов инфраструктуры. Она должна централизованно хранить условия работы детектора и калибровочные данные, разграничивать доступ, связывать данные с проектами и группами и предоставлять средства ввода, просмотра, поиска и экспорта. При этом система должна быть достаточно гибкой, поскольку состав калибровочных и конфигурационных данных будет меняться по мере развития эксперимента и уточнения требований.

В рамках курсовой работы разрабатывается прототип такой базы как веб-приложения. Основное внимание уделяется прикладному уровню: организации проектов, управлению группами доступа, созданию пользовательских таблиц, работе с типизированными колонками и вложенными таблицами, а также разделению прав между администратором

сервиса, администратором группы, редактором и пользователем с правами просмотра. Такой порядок (сначала каркас управления данными, затем специализированные сущности — интервалы валидности, версии, payload) позволяет уже на ранней стадии проверять пользовательские сценарии на работающей системе.

Актуальность работы определяется необходимостью получить расширяемый инструмент, который можно интегрировать в программную экосистему эксперимента SPD и развивать в полноценную Conditions & Calibrations DB. Новизна прототипа в том, что проектная модель доступа, динамическое описание табличных данных, вложенные структуры и проектная заморозка пользователей объединены в одном веб-приложении. Практическая ценность — прототип может быть развернут в контейнерной среде и использован для проверки пользовательских сценариев до перехода к промышленной схеме хранения условий и калибровок.

Дальнейшие разделы построены следующим образом. В разделе постановки задачи формулируются цель, объект, предмет и требования к решению. В обзоре рассматриваются базы условий и калибровок других экспериментов и подходы к управлению доступом. В разделе исследования и решения описываются модель предметной области и архитектура прототипа. Экспериментальная часть посвящена проверке основных пользовательских сценариев.

1. Постановка задачи

Цель работы — разработать прототип базы данных Conditions & Calibrations для эксперимента SPD на коллайдере NICA, обеспечивающий управляемое хранение проектных данных, разграничение доступа и архитектурный задел для последующей работы с условиями, калибровками, интервалами валидности и версиями.

Объект работы — программная инфраструктура хранения и управления условиями реконструкции и калибровочными данными в эксперименте физики высоких энергий. Предмет — веб-приложение, реализующее модель проектов, групп, ролей, пользовательских таблиц и операций управления доступом.

В принятой модели предметной области проект — это изолированное рабочее пространство, к которому администратор сервиса назначает одну или несколько групп. Пользователи получают права через членство в группах и назначенные роли. Администратор группы имеет полный доступ к данным проекта в пределах своей группы, редактор может добавлять и изменять записи, пользователь с правами просмотра — только смотреть и экспортировать данные. Администратор сервиса управляет созданием проекта, его описанием, назначением групп и жизненным циклом проекта, но автоматически право редактировать проектные данные не получает, если не имеет соответствующей роли в назначенной группе.

Система должна поддерживать создание проектов, назначение групп на проект, отображение проектных данных на рабочих страницах и управление участниками проекта. Для повышения безопасности нужен механизм проектной заморозки пользователя: если пользователь заморожен в конкретном проекте, проект должен исчезать из его списка действующих проектов, а прямой доступ к проекту блокироваться. Информация о такой заморозке остается доступной администратору сервиса.

Для хранения пользовательских данных внутри проекта система должна предоставлять механизм создания таблиц. При добавлении колонки

пользователь выбирает тип данных, совместимый с моделью хранения в PostgreSQL. При заполнении строк выполняется проверка корректности вводимых значений, причем ошибки должны отображаться до отправки формы. Дополнительно требуется работа с вложенными таблицами, когда ячейка содержит не простое значение, а ссылку на подтаблицу со своей структурой и собственными записями.

К функциональным требованиям относятся: создание и редактирование проектов; назначение и снятие групп доступа; просмотр участников проекта; заморозка и разморозка участника в пределах проекта; создание таблиц и колонок; добавление, редактирование, удаление и дублирование строк; поддержка вложенных таблиц; экспорт основной таблицы и структуры с подтаблицами; разграничение прав между ролями full access, editor и viewer.

К нефункциональным требованиям относятся: развертывание в Docker, использование PostgreSQL в качестве основной БД, серверная часть на Django с применением SQLAlchemy для доменной модели, расширяемая архитектура (с возможностью добавлять новые страницы и независимые модули), наличие базовых проверок корректности кода и работоспособности основных сценариев.

Для достижения цели нужно решить следующие задачи: проанализировать требования к хранению условий и калибровок; определить модель пользователей, групп, ролей и проектов; спроектировать структуру хранения проектных таблиц и вложенных данных; реализовать серверную логику управления доступом; разработать пользовательский интерфейс для администраторов, редакторов и пользователей просмотра; реализовать проверку вводимых данных и экспорт; провести проверку основных сценариев работы прототипа.

2. Обзор литературы и существующих решений

Цель обзора — определить подходы, которые целесообразно использовать при разработке прототипа Conditions & Calibrations DB для эксперимента SPD. Сравнение существующих решений проводится по нескольким критериям: модель хранения условий и калибровок, поддержка интервалов валидности, работа с версиями и метками, неизменяемость payload-данных, наличие пользовательских инструментов и возможность разграничения доступа между группами пользователей.

В экспериментах физики высоких энергий база условий — это не вспомогательный справочник, а часть вычислительной цепочки реконструкции. Она связывает событие или набор событий с параметрами, действительными на момент регистрации: геометрией детектора, состоянием подсистем, калибровочными константами, alignment-данными и конфигурациями программной обработки. Поэтому обзор существующих решений важен для выделения тех архитектурных принципов, которые имеет смысл закладывать уже в прототип.

2.1. Базы данных условий и калибровок в экспериментах физики высоких энергий

Spin Physics Detector (SPD) – это эксперимент, запланированный для исследования спиновой физики на адронном коллайдере NICA, являющейся мегасайенс-установкой, сооружаемой в Объединённом институте ядерных исследований (ОИЯИ, г. Дубна, Россия). Эксперимент, ориентированный на исследование спиновой структуры протона и дейтрона, а также других спин-зависимых явлений. В публикациях по физической программе SPD отмечается, что установка должна работать с поляризованными пучками и измерять асимметрии в разных каналах реакций. В техническом описании SPD фиксируется сложность детекторной системы и необходимость согласованной работы подсистем, программной реконструкции и вычислительной

инфраструктуры [2]. То есть хранение условий и калибровок должно быть централизованным и воспроизводимым.

В экспериментах LHC широко использовался проект COOL, разработанный как общий компонент LCG Persistency Framework для работы с conditions data. COOL предоставляет программные средства хранения условий в реляционных базах и применялся, в частности, в ATLAS и LHCb [3]. Существенная особенность подхода — отделение событийных данных от условий: события остаются в основной системе хранения, а меняющиеся во времени параметры извлекаются из специализированной базы по времени, тегу или другому идентификатору контекста реконструкции.

В документации LHCb Conditions Database данные условий описываются через иерархию папок, объекты условий, интервалы валидности и теги. Чтобы извлечь корректное значение, недостаточно знать только имя параметра — нужно указать область хранения, момент или интервал времени и версию набора условий [4]. Из этого следует, что условия и калибровки — это не обычные редактируемые записи, а версионируемые данные, привязанные к контексту применения.

Похожая концепция используется в Belle II. В документации basf2 conditions database payload определяется как атомарный объект данных, который связывается с одним или несколькими интервалами валидности, а набор payload и интервалов объединяется в globaltag [5]. Опубликованные наборы неизменяемы: вместо изменения уже использованных данных создаются новые ревизии или новые метки. Это требование напрямую связано с воспроизводимостью физического анализа.

Для рассматриваемой работы перечисленные системы задают целевую модель развития: payload, interval of validity, version или tag и неизменяемость опубликованных данных. Однако на первом этапе прототип решает более базовую инженерную задачу — предоставить пользователям управляемую среду, в которой можно создавать проекты, назначать группы, описывать структуру данных и проверять сценарии доступа. Поэтому в реализации

сделан акцент на проектной модели, динамических таблицах и ролевом интерфейсе, а поддержка IOV и неизменяемых payload рассматривается как следующий слой развития.

2.2. Подходы к управлению доступом и проектными данными

Управление доступом — отдельная часть задачи, потому что разные пользователи эксперимента работают с данными по-разному. Администратор сервиса отвечает за создание проектов и назначение групп, администратор группы управляет данными внутри разрешенного проекта, редактор вносит и исправляет записи, пользователь с правами просмотра может только читать и экспортировать данные. Такая модель близка к role-based access control, но требует учитывать не только роль пользователя, но и его членство в конкретной группе, назначенной на конкретный проект.

Стандартная модель Django включает пользователей, группы, разрешения и сессионную аутентификацию [7]. Эти механизмы удобны как базовый слой идентификации и общего контроля доступа, но напрямую для проектной базы условий их недостаточно. Нужен объектный уровень авторизации: одно и то же действие может быть разрешено в одном проекте и запрещено в другом, если пользователь не входит в назначенную группу или заморожен в рамках конкретного проекта.

Для хранения данных выбран PostgreSQL — он предоставляет развитую систему встроенных типов: целочисленные и вещественные типы, boolean, date, time, timestamp, UUID, text, varchar, numeric, а также JSON/JSONB для полуструктурированных данных [6]. Это позволяет поддерживать пользовательские таблицы с типизированными колонками, не сводя все значения к строкам. Для прототипа важно, что проверка типа выполняется и на пользовательском уровне (до отправки формы), и на серверном уровне при сохранении строки.

Альтернативой могло быть предоставление пользователям прямого доступа к отдельным схемам базы данных, но такой подход усложняет

контроль прав, аудит, валидацию данных и работу с вложенными структурами. Поэтому в прототипе используется прикладная модель: структура пользовательских таблиц хранится в доменных таблицах приложения, а значения строк сохраняются в управляемом формате. Это снижает риск неконтролируемых изменений схемы базы и позволяет реализовать единый интерфейс для всех категорий пользователей.

Отдельным требованием является проектная заморозка пользователя. Она отличается от полной блокировки учетной записи тем, что пользователь сохраняет доступ к системе и к другим проектам, но теряет доступ к конкретному. Такой механизм нужен, когда участник временно исключается из работы над отдельным набором данных, а его учетная запись блокироваться не должна. В результате модель доступа становится более гибкой и соответствует проектной организации работ.

2.3. Выводы к разделу 2

Обзор показывает, что для баз условий и калибровок в экспериментах физики высоких энергий ключевыми являются воспроизводимость, связь данных с интервалами валидности, поддержка версий или меток и неизменяемость опубликованных payload-объектов. Эти принципы используются в существующих системах — COOL, LHCb Conditions Database и Belle II Conditions Database.

Для прототипа Conditions & Calibrations DB эксперимента SPD первоочередная задача — создать прикладной каркас, позволяющий безопасно управлять пользователями, группами, проектами и структурой данных. Такой каркас должен поддерживать ролевое разграничение доступа, динамическое описание таблиц, проверку типов, экспорт и вложенные таблицы. На ту же архитектурную основу затем добавляются специализированные сущности условий: payload, hash-идентификация, intervals of validity, версии и глобальные метки.

3. Исследование и решение задачи

Исходная задача была декомпозирована на несколько взаимосвязанных подзадач: выделение сущностей предметной области, проектирование модели доступа, выбор способа хранения пользовательских таблиц, реализация операций над строками и вложенными таблицами, разработка интерфейса для разных ролей и подготовка контейнерного развертывания. Такая декомпозиция позволяет отделить универсальную часть системы управления проектами от специализированной, связанной с будущей поддержкой payload, интервалов валидности и версий.

Главным проектным решением стало введение проектного рабочего пространства. Проект связывает группу пользователей, набор таблиц и правила доступа. От простой общей базы таблиц это решение отличается тем, что каждая группа работает только с теми проектами, на которые она назначена администратором сервиса. То есть проект становится границей управления данными и одновременно единицей организационного контроля.

Второе важное решение — отказ от автоматического предоставления администратору сервиса права редактировать внутренние данные проекта. Администратор сервиса создает проект, изменяет его название и описание, назначает группы, может заморозить или удалить проект, но работа с данными проекта зависит от роли пользователя в назначенной группе. Такой подход снижает риск административного вмешательства в предметные данные и делает ответственность за содержимое проекта более явной.

3.1. Модель предметной области

Модель предметной области включает пользователей, группы, роли, проекты, участников проектов, таблицы, колонки, строки и вложенные таблицы. Пользователь получает доступ к проекту не напрямую, а через членство в группе, назначенной на проект. Роль пользователя в группе преобразуется в один из уровней доступа: full access, editor или viewer. Full access соответствует администратору группы и дает право управлять схемой

проектных таблиц, editor позволяет изменять записи, viewer ограничивается просмотром и экспортом.

Сущность Project описывает рабочую область. Для нее хранятся название, описание, автор создания, состояние заморозки проекта и служебные поля времени. Связь ProjectGroup фиксирует назначение группы на проект. Такая структура позволяет одному проекту иметь несколько групп, а одной группе участвовать в разных проектах. Это соответствует реальной организации работ, где разные подсистемы и рабочие группы могут иметь пересекающиеся зоны ответственности.

Сущность ProjectUserBlock реализует проектную заморозку пользователя. В отличие от общей блокировки учетной записи, такая заморозка действует только в пределах одного проекта. Если пользователь заморожен в проекте, проект исключается из списка его действующих проектов, а доступ по прямой ссылке блокируется. При этом пользователь сохраняет доступ к другим проектам и к системе в целом. Для администратора сервиса такие проекты остаются видимыми на странице пользователя — это нужно для аудита и восстановления доступа.

Для хранения данных проекта используются сущности ProjectTable, ProjectTableColumn и ProjectTableRow. Таблица принадлежит проекту и может быть корневой или вложенной. Колонка описывает имя, ключ, тип данных, обязательность и порядок отображения. Строка хранит значения в управляемом JSON-представлении. Это позволяет создавать динамические таблицы без выполнения пользовательских DDL-операций над физической схемой PostgreSQL, сохраняя контроль над валидацией и доступом.

Вложенные таблицы моделируются через связь таблицы с родительской строкой и родительской колонкой. Если значение ячейки должно быть сложнее простого скалярного значения, пользователь создает подтаблицу, которая получает собственные колонки и строки. В основной таблице при этом отображается ссылка открытия подтаблицы. Такой вариант удобен для условий и калибровок, где отдельная запись может содержать набор

дополнительных параметров, матрицу коэффициентов или детализированное описание состояния подсистемы.

Типы колонок выбраны с учетом возможностей PostgreSQL: текстовые значения, целые числа разной разрядности, вещественные значения, boolean, date, time, timestamp, UUID, JSONB и специальный тип вложенной таблицы. PostgreSQL предоставляет широкий набор встроенных типов данных [6], поэтому прототип использует типизацию не только как элемент интерфейса, но и как основу серверной проверки значений. Старые обозначения типов приводятся к каноническим (например, integer к int32, double precision к float64).

3.2. Архитектура программного решения

Программное решение построено как веб-приложение на Django. Django используется для маршрутизации, шаблонов, сессий, обработки HTTP-запросов и базового механизма аутентификации [7]. Доменная логика вынесена в отдельные сервисы и репозитории. Этот слой отделяет пользовательский интерфейс от правил доступа и операций с данными, что упрощает добавление новых страниц и сценариев.

Для доменной модели проектов применяется SQLAlchemy ORM. SQLAlchemy позволяет описывать таблицы базы в виде Python-классов и задавать связи между ними [8]. В прототипе это используется для моделей Project, ProjectGroup, ProjectUserBlock, ProjectTable, ProjectTableColumn и ProjectTableRow. Репозиторий ProjectRepository инкапсулирует запросы к базе, проверку доступа и операции создания, обновления, удаления, дублирования и экспорта.

Архитектурно приложение разделено на несколько слоев. Web-слой принимает запросы, извлекает параметры форм и возвращает HTML-страницы или файлы экспорта. Service-слой проверяет высокоуровневые ограничения, связанные с текущим пользователем. Repository-слой выполняет детальные проверки доступа и работу с базой данных. DTO-объекты используются для

передачи данных в шаблоны без раскрытия внутренних ORM-моделей. Такое разделение делает код более устойчивым к росту числа страниц и пользовательских сценариев.

Развертывание выполняется в Docker Compose. Compose-файл описывает PostgreSQL, веб-сервис, мониторинговые компоненты и постоянные тома. Документация Docker Compose определяет такой подход как способ описания сервисов, сетей и томов многоконтейнерного приложения [9]. В данном проекте это позволяет одинаково запускать базу данных и сервер приложения в локальной и тестовой среде, а также сохранять данные PostgreSQL и загружаемые payload-файлы между перезапусками контейнеров.

Пользовательский интерфейс реализован на Django templates и статических CSS/JavaScript-файлах. Для администратора сервиса проекты отображаются карточками, где видны название, описание, назначенные группы и состояние проекта. Страница проекта содержит вкладки данных и участников. Вкладка данных показывает таблицы проекта, форму создания колонок и строк, область редактирования записей и кнопки экспорта. Вкладка участников показывает пользователей, входящих в назначенные группы, их почты, роли и статус заморозки в проекте.

Для удобства работы с таблицами реализована клиентская валидация. При вводе значений поле сразу показывает, соответствует ли оно заданному типу и ограничениям диапазона. Для таблиц условий и калибровок это особенно важно: потеря введенных значений после неудачной отправки формы ухудшает пользовательский опыт и повышает вероятность ошибки. Серверная валидация при этом сохраняется и считается обязательной, поскольку клиентская проверка не относится к доверенным механизмам безопасности.

Экспорт реализован в двух вариантах. Обычный экспорт формирует CSV-представление активной таблицы. Экспорт с подтаблицами формирует JSON-структуру, в которой вложенные таблицы сохраняются рекурсивно. Такой формат подходит для дальнейшей передачи данных в сервисы

обработки или для ручной проверки структуры сложных записей. Дублирование строки также выполняется рекурсивно: если строка содержит подтаблицы, они копируются вместе с их колонками и строками.

3.2.1. Разграничение прав и проектная заморозка пользователей

Разграничение прав построено на вычислении максимального уровня доступа пользователя по группам, назначенным на проект. Если пользователь состоит в нескольких группах проекта, выбирается наиболее высокий уровень доступа. Viewer разрешает просмотр таблиц, участников и экспорт. Editor дополнительно разрешает добавление строк, изменение ячеек, удаление и дублирование записей. Full access разрешает управление схемой таблиц: создание таблиц и колонок, а также открытие вложенных таблиц.

Администратор сервиса имеет отдельный набор прав. Он может создавать проект, изменять его название и описание, назначать группы, удалять проект и управлять жизненным циклом проекта. Однако автоматического права редактировать данные проекта он не получает, если не является администратором назначенной группы. Такое правило введено осознанно: административный доступ к платформе и содержательный доступ к данным эксперимента должны разделяться.

Заморозка проекта и заморозка участника проекта решают разные задачи. Заморозка проекта скрывает весь проект от пользователей, кроме администратора сервиса, и блокирует работу с внутренними данными. Заморозка участника действует точечно: пользователь перестает видеть конкретный проект, остальные проекты остаются доступными. Управлять такой заморозкой могут администратор сервиса и администратор группы с full access в проекте.

На странице пользователя отображаются проекты, в которых он состоит. Если пользователь заморожен в проекте, такой проект отображается только администратору сервиса. Для остальных пользователей он скрывается, что согласуется с правилом отсутствия доступа к замороженному проекту. То есть

одно и то же представление пользователя адаптируется к правам того, кто его просматривает.

Безопасность операций обеспечивается не только скрытием кнопок в интерфейсе, но и серверными проверками в репозитории. Даже если пользователь попытается отправить POST-запрос напрямую, операция будет выполнена только при наличии требуемого уровня доступа. Для веб-приложения такой подход обязателен, поскольку клиентский интерфейс нельзя рассматривать как достаточный механизм защиты.

Заключение

В результате выполнения работы разработан прототип базы данных Conditions & Calibrations для эксперимента SPD на коллайдере NICA. Реализован прикладной каркас: проекты, группы, роли, участники, табличные данные, вложенные таблицы, экспорт и механизмы заморозки доступа. Полученное решение предполагается развивать в сторону полноценной базы условий и калибровок с поддержкой payload, intervals of validity, версий и меток.

В ходе работы предложена модель разграничения доступа, в которой администратор сервиса управляет проектами и назначением групп, а управление данными проекта зависит от роли пользователя в назначенной группе. Такая модель разделяет административные функции платформы и содержательную ответственность за данные эксперимента. Дополнительно реализована проектная заморозка пользователя — она ограничивает доступ только к конкретному проекту без полной блокировки учетной записи.

Практическая часть включает реализацию динамических пользовательских таблиц с типизированными колонками, проверкой данных, редактированием строк, удалением, дублированием и поддержкой вложенных таблиц. Для удобства пользователей добавлена клиентская валидация, предотвращающая потерю введенных значений при ошибках заполнения. Для обмена данными реализован экспорт основной таблицы в CSV и экспорт структуры с подтаблицами в JSON.

Разработанное приложение может быть развернуто в контейнерной среде Docker с использованием PostgreSQL. Архитектура разделена на web-, service- и repository-слои, что упрощает масштабирование проекта и добавление новых страниц для разных категорий пользователей. Базовые проверки кода и сценариев подтверждают работоспособность реализованного прототипа.

Дальнейшее развитие системы предполагает добавление специализированных сущностей Conditions & Calibrations DB: неизменяемых

payload-объектов, hash-идентификации, intervals of validity, глобальных тегов, истории публикации и REST-интерфейса для интеграции с программной реконструкцией. То есть текущий прототип является основой для перехода от универсального управления проектными данными к полноценной инфраструктуре хранения условий и калибровок эксперимента SPD.

Литература

1. Guskov A., Datta A., Karpishkov A., Denisenko I., Saleev V. Probing Gluons with the Future Spin Physics Detector // Physics. 2023. Vol. 5, № 3. P. 672–687. DOI: 10.3390/physics5030044.
2. Technical Design Report of the Spin Physics Detector at NICA [Электронный ресурс]. arXiv:2404.08317. URL: https://arxiv.org/abs/2404.08317 (дата обращения: 12.05.2026).
3. Valassi A. COOL, LCG Conditions Database for the LHC Experiments [Электронный ресурс]. CERN Document Server, 2008. URL: https://repository.cern/records/0pafy-n1p34 (дата обращения: 12.05.2026).
4. LHCb Conditions Database Graphical User Interface [Электронный ресурс]. URL: https://lhcb.web.cern.ch/computing/Frameworks/DetCond/documents/conddbui.pdf (дата обращения: 12.05.2026).
5. Belle II Software Documentation. Conditions Database [Электронный ресурс]. URL: https://training.belle2.org/framework/doc/conditions-database.html (дата обращения: 12.05.2026).
6. PostgreSQL Global Development Group. PostgreSQL 16 Documentation. Chapter 8. Data Types [Электронный ресурс]. URL: https://www.postgresql.org/docs/16/datatype.html (дата обращения: 12.05.2026).
7. Django Software Foundation. User authentication in Django [Электронный ресурс]. URL: https://docs.djangoproject.com/en/6.0/topics/auth/ (дата обращения: 12.05.2026).
8. SQLAlchemy. SQLAlchemy 2.0 Documentation. ORM [Электронный ресурс]. URL: https://docs.sqlalchemy.org/20/orm/ (дата обращения: 12.05.2026).
9. Docker Inc. Docker Compose Documentation [Электронный ресурс]. URL: https://docs.docker.com/compose/ (дата обращения: 12.05.2026).