ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

Научная статья

УДК 519.683
DOI: 10.18101/2304-5728-2022-1-3-17

АРХИТЕКТУРА ИЕРАРХИЧЕСКИ МОДИФИЦИРУЕМО-ПЕРЕСЕКАЮЩЕЙСЯ БАЗЫ ДАННЫХ БИОЭКОЛОГИЧЕСКИХ ПАРАМЕТРОВ

© **Аргучинцев Александр Валерьевич**
доктор физико-математических наук, профессор,
заведующий кафедрой вычислительной математики и оптимизации,
Иркутский государственный университет
Россия, 664003, г. Иркутск, ул. К. Маркса, 1
arguch@math.isu.ru

© **Кедрин Виктор Сергеевич**
кандидат технических наук, доцент,
Иркутский государственный университет
Россия, 664003, г. Иркутск, ул. К. Маркса, 1
kedrinvs@isu.ru

© **Кедрина Мария Сергеевна**
магистрант,
Иркутский государственный университет
Россия, 664003, г. Иркутск, ул. К. Маркса, 1
kedrinams@isu.ru

Аннотация. Рассматривается архитектурная модель научно-исследовательской платформы BaikalIntelli. Платформа предназначена для универсального хранения, ввода и обработки произвольных пространственно-временных баз данных, содержащих биоэкологические параметры. Гибкая система дает возможность работать с различной структурой таблиц и составом реквизитов. Предложены информационные технологии, позволяющие реализовать базовую функциональность универсального метамеханизма систематизации данных. Используемый архитектурный паттерн реализует логику изолирования модели данных от представления и обеспечивает автономные принципы разработки механизмов платформы BaikalIntelli, нацеленные на снижение повторения элементов программного кода в программных классах платформы. Механизм платформы апробирован с помощью конкретной базы данных «Пространственно-временная изменчивость популяций олигохет в районе сброса промышленных очищенных сточных вод Байкальского целлюлозно-бумажного комбината». Спроектированные механизмы платформы позволяют обеспечить надежное хранение и оперативное извлечение данных наблюдений.

Ключевые слова: исследовательская платформа, клиент-серверная архитектура, паттерны хранения данных, базы данных, полиморфная информация, систематизация данных, иерархические модели, динамический интерфейс, веб-платформа, динамика популяций.

Благодарности

Проект реализуется победителем конкурса «Академический десант» благотворительной программы «Стипендиальная программа Владимира Потанина» Благотворительного фонда Владимира Потанина, договор гранта № ГСАД-0022/21. Исследование также выполнено при финансовой поддержке РФФИ и Правительства Иркутской области в рамках научного проекта № 20-41-385002.

Для цитирования

Аргучинцев А. В., Кедрин В. С., Кедрина М. С. Архитектура иерархически модифицируемо-пересекающейся базы данных биоэкологических параметров // Вестник Бурятского государственного университета. Математика, информатика. 2022. № 1. С. 3–17.

Введение

Применение математических моделей для исследования динамики популяций, описываемых дифференциальными и интегро-дифференциальными уравнениями, сопряжено с проработкой комплекса смежных вопросов по организации оптимального программного механизма для формирования цифровых процессов хранения, обработки и анализа большого объема гетерогенной информации. Разрозненность имеющихся данных, плохая структурированность и недостаточная их приспособленность для обработки современными математическими методами диктуют необходимость развития комплексного подхода к организации информационной исследовательской среды, которая позволяет решить проблемы жесткой структурированности и анализа взаимосвязей в отличие от стандартных механизмов организации научно-исследовательской информации (MS Excel, MS Access и т. д.).

При реализации информационно-аналитической платформы, ориентированной на процессы исследования динамики популяций, необходимо решение следующих ключевых проблем:

1. Определение универсальных архитектур хранения данных, позволяющих аккумулировать пространственно-временные базы данных различного рода.
2. Создание программных механизмов оптимального представления и систематизации данных большой размерности, а также процессов их эффективной визуализации в табличном и графическом представлениях.
3. Синхронизация информации в условиях дистанционного взаимодействия участников в среде интернет с учетом различных многообразных вариантов пользовательских интерфейсов.

Целью разработки платформы BaikalIntelli является описание фундаментальных прототипов информационных архитектурных моделей и алгоритмов, ориентированных на создание программно-технических средств,

позволяющих сформировать среду проектирования и учета математических моделей для исследования популяций байкальских растений и бентосных организмов. В этой связи с позиции системного подхода были выделены следующие ключевые задачи:

- создание программных средств описания и алгоритмизации математического аппарата для решения дифференциальных уравнений с частными производными и интегро-дифференциальных уравнений;
- формирование программных архитектур для описания механизмов хранения и обработки пространственно-временных баз данных разнородного характера с целью обеспечения комплексного описания процессов в сложных биологических сообществах, состоящих из большого количества видов;
- организация интерфейсов и функционала для проведения предварительных статистических расчетов по идентификации параметров моделей;
- обеспечение когнитивного процесса принятия решений [1] для выбора адекватных вариантов из достаточно широкого набора расчетных вычислительных моделей;
- разработка и применение алгоритмических методов оптимального управления для идентификации параметров моделей на основе множества векторных серий оцифрованных реальных данных.

Термин Intelli, заложенный в названии платформы BaikalIntelli, предложен с учетом требований к современным агрегатным системам бизнес-анализа. Разрабатываемый прототип платформы должен обеспечивать решение следующих задач:

- аналитика, в том числе анализ разнородных данных большой размерности, статистический анализ;
- сотрудничество специалистов разных профилей;
- управление знаниями;
- измеримость и отчетность, то есть получение необходимых избранных и сводных данных по задаваемым критериям.

Проектирование научно-исследовательской платформы BaikalIntelli основано на элементах эффективной программной архитектуры [2; 3], которая позволяет определять комплекс распределенных механизмов для

хранения и обработки информации. В этой связи уникальность платформы заключается в разработке полиморфного универсального метамеханизма систематизации с целью внесения и обработки произвольных пространственно-временных баз данных с различной структурой таблиц и составом реквизитов. Это определяет возможности организации интеграционного взаимодействия исследователей различных областей естественных наук для изучения механизмов антропогенного воздействия на оз. Байкал, а также формирование возможностей применения современных математических, статистических и информационных методов. При этом для обеспечения гибкости программные механизмы платформы направлены на организацию взаимодействия участников через создание защищенных веб-интерфейсов, позволяющих организовать целевое рабочее пространство.

1 Архитектурный паттерн

Архитектурный паттерн MVC позволяет реализовать логику изолирования модели данных от представления и обеспечить автономные принципы разработки механизмов платформы BaikalIntelli, нацеленные на снижение повторения элементов программного кода в программных классах платформы в соответствии с принципом Don't repeat yourself (DRY) [1]. Также описываемый паттерн позволяет придерживаться принципов разграничения программных слоев и их целесообразности. При этом слои приложения общаются между собой в рамках интерфейсных методов взаимодействия классов в каждом слое логики. При разработке прототипа проектной модели универсального метамеханизма хранения полиморфных данных проведено расширение применения паттерна архитектуры MVC. Определена раздельная программная логика, которая позволяет не только систематизировать классы и объекты движка в рамках указанной иерархии логических сущностей проектирования, но и создавать механизм настраиваемой группировки, который формирует для представления только те данные, которые действительно необходимы для отображения конкретному пользователю системы. К примеру, если данные исследователя в базе данных занимают десятки тысяч значений, то для отображения на клиенте в актуальном виде нужно всего лишь несколько десятков значений (в зависимости от области видимости виджета графического интерфейса). Поэтому в актуальный момент времени контролер от уровня модели может передать представлению только необходимые значения, что позволит повысить скорость обработки данных на уровне визуализации интерфейса для больших наборов данных и уменьшить потребности в памяти при графической визуализации по сравнению с подходом, требующим считывания всех данных. Еще одним преимуществом архитектуры MVC является то, что она ориентирована на решение проблемы синхронизации данных для разных представлений. Так, связывая

одну модель с двумя или более представлениями, можно представить взаимодействие данных с позиции их отображения различными способами. Благодаря этому изменения в данных будут синхронно отражаться во всех представлениях, связанных с моделью.

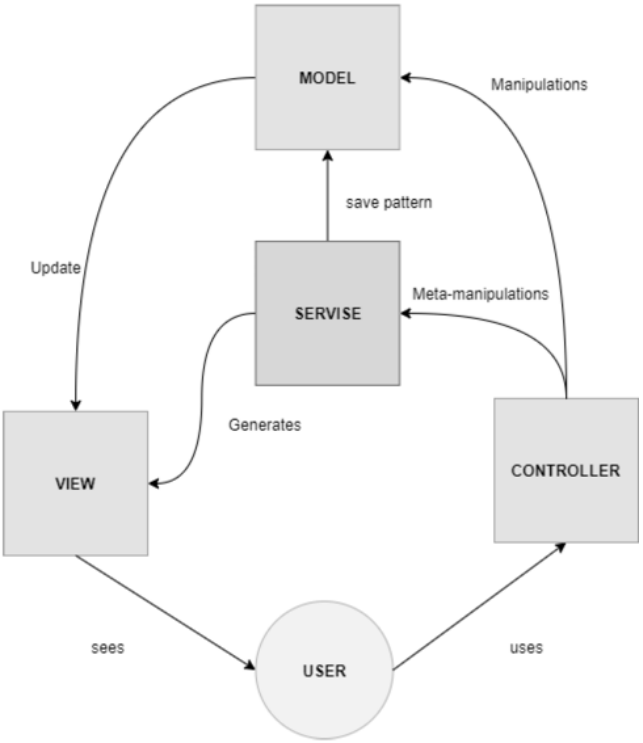

Рис. 1. Архитектурный паттерн Model-Sevise-View-Controller

Паттерн MVC (Model-Service-View) подразумевает, что вся логика приложения реализуется в контроллере, который в случае разработки достаточно сложной клиентской и аналитической частей описывает не только методы передачи данных, но и бизнес-процессы, заложенные в интерфейсе пользователя и логике доступа к модели данных. Это нарушает The Single Responsibility Principle (SRP) [2] в классах контроллера, а также значительно перегружает код в данной части контроллера и может спровоцировать определенные трудности при масштабировании. В этой ситуации необходимо выделить отдельный уровень сервис-слоя (Service), реализующий описание бизнес-логики приложения. Такая интерпретация подразумевает разделение программных модулей на четыре основных блока (рис. 1):

1. Модель (Model).
2. Сервис-слой (Service).
3. Контроллер (Controller).
4. Представление (View).

Добавление нового блока Service с позиции архитектуры MSVC (Model-Service-Controller-View) позволяет разделить функциональность контроллера на две части:

- взаимодействие моделей с представлениями и вызов сервиса описываются с помощью контроллера;
- взаимодействие с металогикой преобразования универсальных структур данных на уровне операций моделями, а также описание блоков интерфейса реализуют сервис-слой.

Внедрение нового слоя Service влияет и на реализацию паттерна MVC. Теперь взаимодействие между блоками в MSVC подразумевает вызов стороннего контроллера сервисом. Таким образом, возможно расширение паттерна MVC до HMSVC (Hierarchical Model Service View Controller), когда структура описания контроллера может быть иерархической, что позволит значительно повысить возможности повторного использования кода и увеличит его способность к масштабированию и определению метафункционала. В результате сформированный паттерн Model-Service-View-Controller (далее — MSVC) позволяет не только отделять данные от представления, но и описывать сущность метаслоя для оперирования конструкциями произвольной иерархии данных, а также определять функционал динамического формирования интерфейса на клиенте. При этом Service слой одновременно присутствует в серверной и клиентской частях платформы.

На сервере:

- описываются механизмы конвертации данных из уровня хранения в систематизацию в виде табличных объектов для вывода в интерфейс клиента;
- определяются механизмы целостности хранения структур в полях базы данных в формате JSON [3];
- создается логика описания паттернов динамических клиентских интерфейсов;
- формируется асинхронный механизм исполнения скриптов обработки данных и расчета моделей.

На клиенте:

- формируется движок генерации (рендеринга) элементов графического интерфейса;
- создается компоновка табличных данных;
- реализуются механизмы компоновки вывода обработанных данных;
- определяется программный функционал визуализации отчетности для скриптов обработки данных.

В клиентской части представление (View) отвечает за формирование логики графического интерфейса пользователя (UI). В этом случае клиентский сервис-слой ориентирован на вызов контроллера сервера, который передает метаданные, реализующие паттерны бизнес-логики организации компоновки и создания динамического интерфейса представления, описания и обработки полиморфных данных для пользователя непосредственно на клиенте, используя скрипты на языке программирования JavaScript. Поэтому расширение серверной логики приложения организует хранение и использование следующих сущностей:

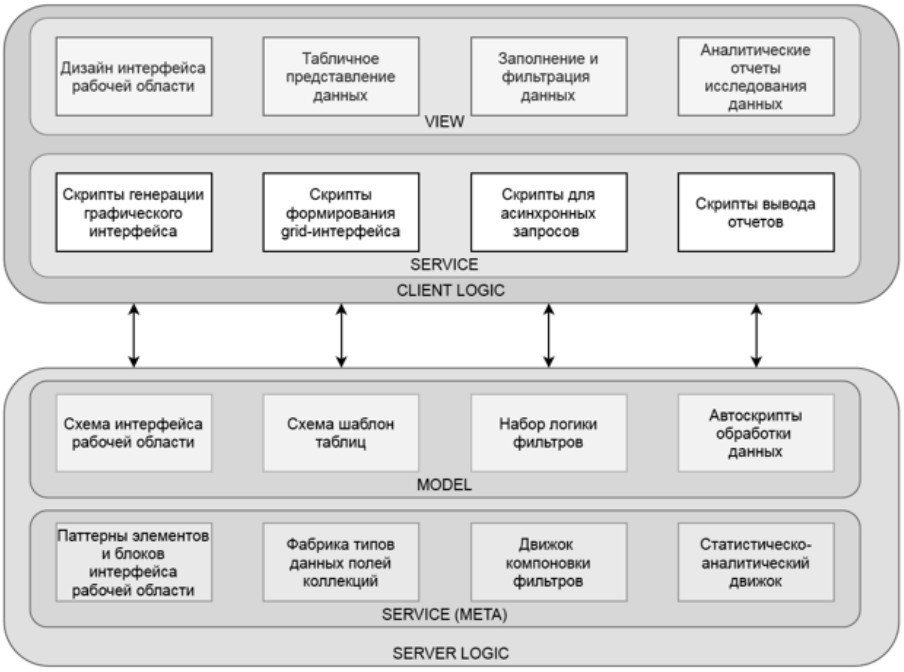

Рис. 2. Архитектурный паттерн динамических интерфейсов и компоновки данных

- аналитические пользовательские скрипты, определяемые пользователями в рамках процесса обработки данных и проектирования математических моделей;
- модули библиотек, формирующие набор самодостаточных пакетов, содержащих комплекс функций для работы с математическими объектами при проектировании расчетных алгоритмов, предназначенных для обработки пространственно-временных данных;
- микроприложения, представляющие набор программных объектов для выделения логических механизмов платформы, организованных по архитектуре MVC.
- библиотеки интерфейсов, содержащие описание компонентов виджетов, предназначенных для формирования сложных логических блоков представления UI.

Клиент-серверная архитектура [4] решения априори предполагает разделение программно-технологической схемы на два функциональных контура, ориентированных на:

- архитектуру программных интерфейсов в рамках реализации программных скриптов для исполнения в браузерных клиентах пользователей;
- логику обработки и управления данными на уровне серверного взаимодействия с СУБД [1].

Введение в модель серверной логики универсального метамеханизма хранения и систематизации полиморфных данных позволяет сформировать паттерн архитектурной модели (рис. 2), реализующий логику кастомизации интерфейсов организации и вывода данных. Такой концептуальный подход формирования динамических интерфейса позволит управлять:

- произвольным конструированием блоков рабочей области исследователя в рамках научной секции данных;
- созданием произвольных табличных представлений серий данных;
- организацией элементов интерфейса фильтра полей данных для интерактивного формирования срезов и выборок данных, а также агрегирования информации в пространственно-временных базах данных в соответствии с предъявляемыми требованиями;
- пользовательскими механизмами обработки данных.

[1] ГОСТ Р ИСО МЭК ТО 10032-2007: Эталонная модель управления данными. М.: Стандартинформ, 2009. 45 с.

Таким образом, предложенная архитектура MSCV позволяет значительно детализировать компонентную разработку платформы с позиции выделения аналитического механизма, повышает способность кода к повторному использованию и решает проблемы, связанные с реализацией разнородной логической структуры как всего приложения, так и отдельных его частей, а также выделяет сущности для описания металогики хранения и систематизации полиморфных данных.

2 Модель хранения полиморфной информации в базе данных

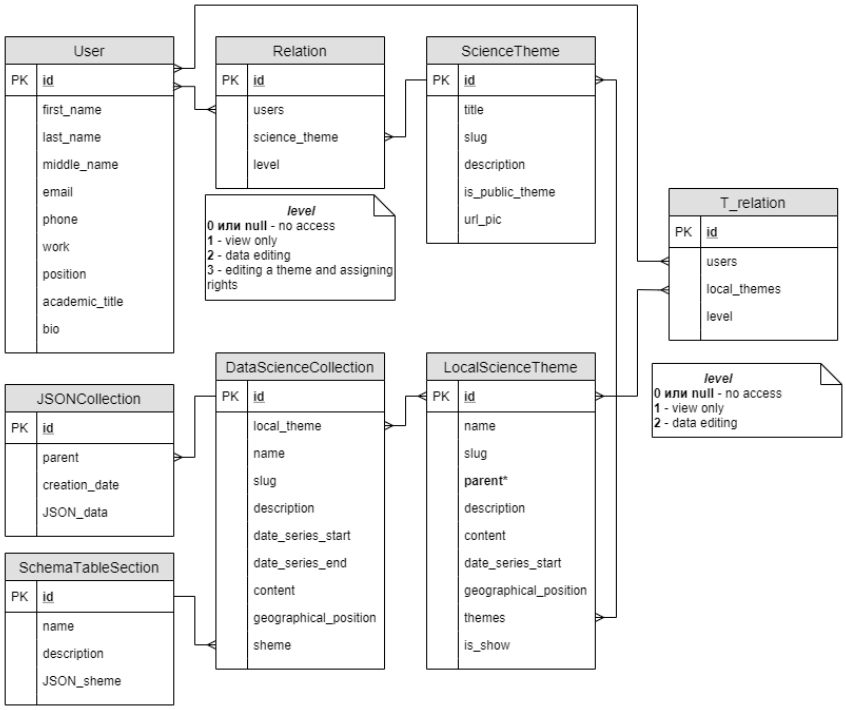

Рис. 3. ER-диаграмма организации данных платформы BaikalIntelli

Платформа BaikalIntelli позволяет организовать хранение, внесение и обработку произвольных пространственно-временных баз данных с различной структурой таблиц и составом реквизитов. Описываемая модель базы данных позволяет комплексно решать задачи сбора и обработки пространственно-временных данных для оценки распределенных параметров при проектировании вычислительных алгоритмов расчета математических моделей динамики популяций. Для этого в структурной логике концептуальной модели организации данных платформы BaikalIntelli (рис. 3) заложена реализация следующих функциональных возможностей:

- организация ввода данных в разрезе глобальных направлений и вложенных научных секций;
- древовидная архитектура научных секций в пределах направления с помощью организации иерархической структуры хранения данных;
- произвольные схемы шаблонов таблиц в рамках научной секции, описываемые с помощью формата JSON;
- ввод произвольного количества таблиц в пределах научной секции с помощью универсального механизма хранения данных в формате JSON;
- привязка к географическому положению в пределах секции и таблицы сбора данных (предполагается, что в таблице данных в пределах секции более детальное местоположение);
- возможность описания секции и таблицы с помощью MarkDown формата;
- система доступа зарегистрированных пользователей к данным в рамках выбранного направления и секции;
- универсальный механизм хранения данных в формате JSON;
- распределенная модель REST-обмена между клиентской и серверной частями приложения.

Разработанная схема организации данных, обеспечивающая серверную логику механизмов хранения пользовательских баз данных в рамках платформы BaikalIntelli, состоит из следующих моделей, реализованных на ORM-уровне архитектуры приложения:

1. «ScienceTheme» — модель данных, предназначенная для формирования перечня и описания глобальных научных направлений (проектов) исследований.
2. «LocalScienceTheme» — иерархическая модель данных, предназначенная для описания локальных научных секций в пределах проекта, связанного с местом проведения исследований, и позволяющая организовать логическую древовидную структуру пользовательских баз данных.
3. «SchemaTableSection» — модель данных, предназначенная для универсального описания схемы хранения табличных данных с помощью формата JSON.
4. «DataScienceCollection» — модель данных, предназначенная для описания таблиц пользовательских баз данных исследовательских данных в рамках заданной схемы хранения табличных данных с помощью модели «SchemaTableSection».

5. «JSONCollection» – модель данных, предназначенная для хранения отдельных записей таблиц пользовательских баз данных в рамках заданной схемы хранения табличных данных с помощью формата JSON.

3 Структурная модель организации полиморфных данных в формате JSON

Для реализации модели организации полиморфных данных предложена представленная на рис. 4 схема моделей хранения данных, состоящая из трех связанных таблиц.

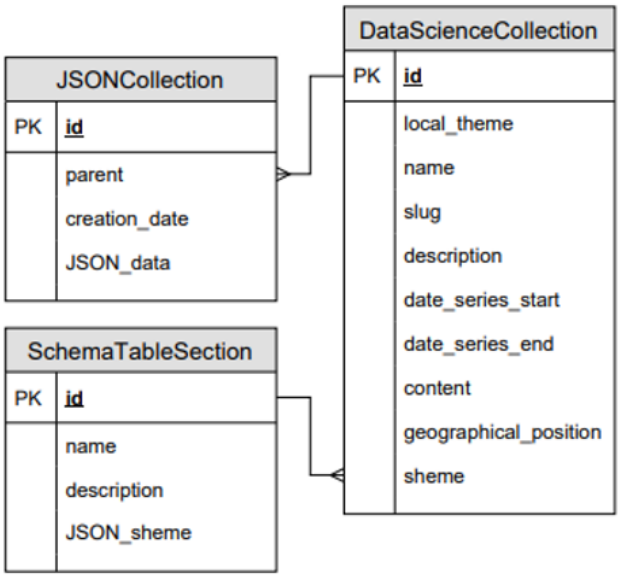

Рис. 4. ER-диаграмма структуры организации полиморфных данных

Модель DataScienceCollection предназначена для описания свойств произвольной пространственно-временной таблицы. При этом для описания пространственно-временной таблицы используются следующие реквизиты:

- name (CharField, max_length=150) — имя эксперимента по сбору данных;
- slug (SlugField, max_length=150) — уникальная строка-идентификатор, понятная человеку;
- description (MDTextField) — описание таблицы коллекции;
- date_series_start — дата начала сбора данных;

- date_series_end — дата окончания сбора данных;
- content (MDTextField) — расширенное описание таблицы;
- geographical_position (ManyToManyField geographical) — уточненное географическое местоположение;
- schema (ForeignKey SchemaTableSection) — связь со схемой аттрибутов коллекции.

Модель ShemaTabaleSection предназначена для формирования схемы хранения коллекции и содержит следующие реквизиты:

- name (CharField, max_length=150) — имя схемы коллекции;
- description(MDTextField) — описание схемы коллекции;
- JSON_sheme(TextField) — JSON-кортеж, описывающий схему коллекции.

Модель данных «SchemaTableSection» предназначена для универсального описания схемы хранения табличных данных с помощью формата JSON. В самом простом варианте структура JSON-кортеж для описания схемы состоит из массива вложенных словарей, содержащих ключи:

- name[I] — имя реквизита;
- type[I] — тип реквизита;
- column[I] — номер колонки.

Модель JSONCollection хранит экземпляры записей в рамках заданного шаблона схемы, описываемой в моделе ShemaTabaleSection. Модель JSONCollection определяется реквизитами:

- table (ForeignKey DataScienceCollection) — связь с таблицей эксперимента;
- id — идентификатор строки;
- date — дата записи;
- JSON_attribute(TextField) — JSON-кортеж в соответствии со схемой атрибутов в таблице эксперимента.

Вид JSON-кортежа для хранения экземпляров записи определяется массивом вложенных словарей, состоящих из ключей:

- name[I] — имя реквизита;
- value[I] — значение реквизита.

В рамках модели LocalScienceTheme реализован механизм формирования древовидного перечня записей на базе технологии иерархической организации данных (modified-preorder-tree-traversal [5]) на уровне модели данных. Данный механизм позволяет:

- формировать рекурсивный иерархический перечень на уровне отношений «родитель — потомок»;
- рекурсивно обходить построенное дерево данных;
- добавлять узлы в дерево данных;
- обновлять узлы в дереве данных;
- обеспечивать каскадное удаление ветвей дерева и зависимых данных.

На уровне моделей SchemaTableSection и JSONCollection реализован механизм универсального хранения данных. Ключевой особенностью организации данных моделей является наличие особого типа полей и механизмов работы с форматом JSON, который позволяет иерархически описывать данные в виде комбинаций

- коллекций пар ключ/значение, которые на уровне языков программирования могут быть интерпретированы как структуры, словари;
- упорядоченных списков значений, которые на уровне языков программирования могут быть интерпретированы типами данных: «массив», «список» или «последовательность».

Заключение

Таким образом, в рамках первого этапа разработки платформы BaikalIntelli, учитывая описанные технологии, были получены следующие результаты.

1. Спроектирована и реализована серверная логика универсального метамеханизма систематизации данных в соответствии с концептуальной ER-моделью с помощью веб-фреймворка Django 3.0.
2. Спроектирована и заполнена пользовательская база данных «Пространственно-временная изменчивость популяций олигохет в районе сброса промышленных очищенных сточных вод Байкальского целлюлозно-бумажного комбината» с разбивкой относительно участков сбора данных (на базе платформы BaikalIntelli).
3. Реализованы механизмы фильтрации данных на базе ORM технологии.
4. Реализован первоначальный табличный веб-интерфейс клиентской части вывода пользовательской базы данных «Пространственно-временная изменчивость популяций олигохет в районе сброса промышленных очищенных сточных вод Байкальского целлюлозно-бумажного комбината» (библиотека интерфейсов Bootstrap, язык программирования JavaScript).

Спроектированные механизмы BaikalIntelli позволяют обеспечить механизмы надежного хранения и оперативного извлечения данных наблюдений популяций олигохет с применением разной иерархической систематизации с позиции факторных признаков и накладываемых условий отбора информации. Интерфейс базы данных ориентирован на получение данных с использованием глобальной сети интернет (REST API) в csv формате в рамках решения задач анализа данных и обеспечения расчетов вычислительных моделей в области математического моделирования динамики популяций для формирования частных критериев оценки антропогенного воздействия на экологическую обстановку оз. Байкал.

Литература

1. Ruby S., Copeland D. B., Thomas D. Agile Web Development with Rails 6. The Pragmatic Programmers, LLC, 2020. 494 p.
2. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Boston: Pearson Education, 2008. 431 p.
3. Crockford D. The application/json Media Type for JavaScript Object Notation (JSON). Internet Engineering Task Force, 2020. URL: https://www.rfc-editor.org/rfc/rfc4627.html (дата обращения 01.12.2021).
4. Коржов В. Многоуровневые системы клиент-сервер. URL: https://www.osp.ru/nets/1997/06/142618 (дата обращения 01.12.2021). Текст: электронный.
5. Tulder V. G. Storing Hierarchical Data in a Database. URL: https://www.sitepoint.com/hierarchical-data-database (дата обращения 01.12.2021).

Статья поступила в редакцию 25.11.2021; одобрена после рецензирования 04.03.2022; принята к публикации 15.03.2022.

ARCHITECTURE OF A HIERARCHICALLY MODIFIABLE-INTERSECTING DATABASE OF BIOECOLOGICAL PARAMETERS

Alexander V. Arguchintsev
Dr. Sci. (Phys. and Math.), Professor,
Irkutsk State University
1 Karl Marx St., Irkutsk 664003, Russia
arguch@math.isu.ru

Victor S. Kedrin
Cand. Sci. (Engineering), A/Prof,
Irkutsk State University
1 Karl Marx St., Irkutsk 664003, Russia
kedrinvs@isu.ru

Maria S. Kedrina
Assistent,
Irkutsk State University
1 Karl Marx St., Irkutsk 664003, Russia
kedrinams@isu.ru

Abstract. The architectural model of the BaikalIntelli research platform is considered. The platform is designed for universal storage, input and processing of arbitrary spatial and temporal databases containing bioecological parameters. The flexible system makes it possible to work with different table structure and composition of the requisites. Information technologies are proposed to implement the basic functionality of the universal metamechanism of data systematization. The used architectural pattern implements the logic of isolating the data model from the representation. It provides the autonomous principles of development of the BaikalIntelli platform mechanisms aimed at reducing repetition of elements of program code in the program classes of the platform. The platform mechanism is tested using the specific database "Spatial and temporal variability of oligochaete populations in the area of discharge of industrial treated wastewater of the Baikal pulp and paper mill". The designed platform mechanisms allow to provide mechanisms for reliable storage and rapid retrieval of observation data.

Keywords: research platform, client-server architecture, data storage patterns, databases, polymorphic information, data systematization, hierarchical models, dynamic interface, web platform, population dynamics.

For citation
Arguchintsev A. V., Kedrin V. S., Kedrina M. S. Architecture of a Hierarchically Modifiable-Intersecting Database of Bioecological Parameters // Bulletin of Buryat State University. Mathematics, Informatics. 2022. N. 1. Pp. 3–17.

The article was submitted 25.11.2021; approved after reviewing 04.03.2022; accepted for publication 15.03.2022.