УДК 38

Адаптивная архитектура программно-аппаратного комплекса хранения и обработки данных

Костиков Юрий Александрович
Кандидат физико-математических наук,
заведующий кафедрой 812,
Московский авиационный институт
(национальный исследовательский университет),
125993, Российская Федерация, Москва, Волоколамское шоссе, 4;
e-mail: jkostikov@mail.ru

Павлов Виталий Юрьевич
Кандидат физико-математических наук,
заведующий кафедрой 318,
Московский авиационный институт
(национальный исследовательский университет),
125993, Российская Федерация, Москва, Волоколамское шоссе, 4;
e-mail: kafedra@istmati.ru

Романенков Александр Михайлович
Кандидат технических наук, доцент,
кафедра 812,
Московский авиационный институт
(национальный исследовательский университет),
125993, Российская Федерация, Москва, Волоколамское шоссе, 4;
e-mail: romanaleks@gmail.com

Терновсков Владимир Борисович
Кандидат технических наук, доцент,
кафедра 318,
Московский авиационный институт
(национальный исследовательский университет),
125993, Российская Федерация, Москва, Волоколамское шоссе, 4;
e-mail: vternik@mail.ru

Аннотация

В данной статье описывается методика разработки программного комплекса хранения, обработки и подготовки данных для построения математических моделей и сопоставления результатов математического моделирования с результатами испытаний и экспериментов. Предлагаемый авторами программный комплекс отличается гибкой модульной архитектурой, включающей как банк данных, так и разного рода интерфейсы для взаимодействия с этими данными и их конвертации в необходимые форматы. Состав и алгоритмы взаимодействия модулей могут меняться в зависимости от типов данных и решаемых задач. Рассматриваются существенные аспекты, на которые нужно обращать внимание при разработке такого рода программных продуктов. Разработанная архитектура программного комплекса может быть применена практически во всех отраслях экономики для решения экономико-статистических, научно-технических и организационно-технологических задач, предусматривающих построение феноменологических математических моделей и/или сопоставление экспериментальных данных с результатами математического моделирования.

Для цитирования в научных исследованиях

Костиков Ю.А., Павлов В.Ю., Романенков А.М., Терновсков В.Б. Адаптивная архитектура программно-аппаратного комплекса хранения и обработки данных // Экономика: вчера, сегодня, завтра. 2017. Том 7. № 9А. С. 192-207.

Ключевые слова

Программный комплекс, хранение данных, обработка данных, математическая модель, интерфейс, база данных, WPF, ASPCore. netframework 4.6.

Введение

В различных отраслях экономики постоянно возникает потребность в обработке экспериментальных данных, причем данные могут быть совершенно произвольной природы и структуры, например экономико-статистические данные, результаты опроса какой-либо группы людей, замеры какой-либо физической величины, характеристики изделия или условий, в которых эксплуатируется технический объект и работают сотрудники, и др. Очевидно, что необходимо обеспечить хранение не только самих данных, но и их атрибутов: информации о времени и месте сбора данных, проведении испытаний и замеров, составе участников опросов

и экспериментов, конфигурации оборудования, методике измерений и многом другом. В результате объемы данных, необходимых для хранения и обработки, очень быстро возрастают и возникает ряд трудностей с их систематизацией и использованием. Стоит также отметить, что на практике часто отсутствуют жестко стандартизированные форматы входных данных, что значительно усложняет процесс внесения исходных данных в компьютерную систему. Для решения подобного рода задач разработан программный комплекс с гибкой архитектурой и модульной структурой, позволяющей адаптировать его для хранения и обработки разнообразных типов данных.

Описание разработанного программно-аппаратного комплекса

Введем термины, которыми будем описывать внутреннее устройство разработанного программного комплекса. Общая структура комплекса включает следующие компоненты: серверную часть (Server), функциональную часть (Processor), представительную часть (Viewer) и часть конвертации данных (Importer).

Программный комплекс представляет собой классическую клиент-серверную многопользовательскую информационную систему, реализованную с использованием современных компьютерных технологий. Хранилище данных является реляционной базой данных (далее – БД). В качестве системы управления базами данных (далее – СУБД) возможно использовать MySQLServer, MS SqlServer, Oracle DB, PostgreeSQL и другие. В данном случае используется PostgreeSQL. Взаимодействие с СУБД происходит не непосредственно, а с помощью промежуточного уровня (Provider), с которым, в свою очередь, связан уровень API.

Уровень API предоставляет средства для доступа к данным. Кроме того, на данном уровне обеспечиваются разграничение прав доступа, а также все возможные проверки безопасности для соответствующих запросов. С уровнем API взаимодействуют остальные части программного комплекса (Viewer, Proccessor, Importer).

Одной из особенностей клиент-серверной архитектуры приложения является то, что непосредственный доступ к самой БД имеет лишь одна «доверенная» машина – та, на которой установлено серверное приложение (Server), и все операции над существующими данными проходят через уровень авторизации и защиты прав доступа, что исключает необходимость постоянного шифрования данных и ускоряет работу клиентских приложений. В свою очередь, каждое клиентское приложение будет решать строго определенную задачу. Схема предлагаемой взаимосвязи модулей представлена на рисунке 1.

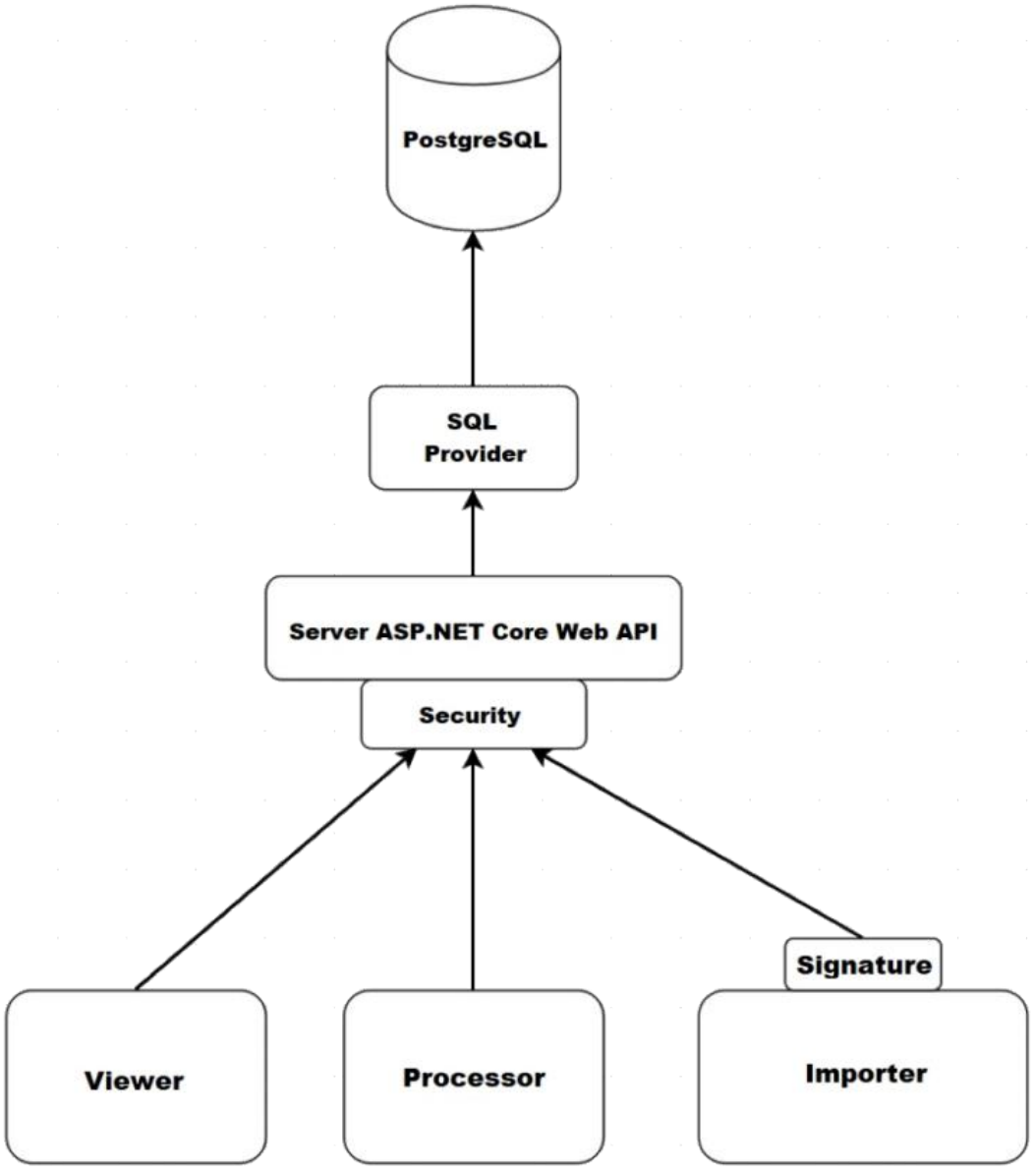

Рисунок 1 - Схема взаимосвязи модулей

Processor – клиентское приложение для обработки данных. В нем сосредоточены все алгоритмы обработки данных и визуализации данных, например построения графиков, механизмы импорта и экспорта данных. К данному приложению имеют доступ две группы пользователей: администраторы и разработчики.

Viewer – клиентское приложение для просмотра графических материалов, построенных по экспериментальным данным. Оно может быть установлено на машину малой мощности, что позволяет значительно выиграть в стоимости рабочих мест сотрудников.

Importer – клиентское приложение для импорта тестовых данных, полученных, например, в

виде файла Microsoft Excel. Задача приложения – гарантировать подлинность импортированных данных, которые впоследствии будут подписаны открытым ключом пользователя.

Для доступа к модулям Viewer и Importer права администратора или разработчика не требуются.

В качестве средств разработки были выбраны библиотеки Windows Presentation Foundation (WPF) [Мак-Дональд, 2015] и технология разработки серверных приложений ASP.NET Core от Microsoft.

WPF представляет собой обширный API-интерфейс для создания настольных графических программ, имеющих насыщенный дизайн и интерактивность. В рамках технологии WPF возможна быстрая и гибкая настройка пользовательского интерфейса под нужды пользователя.

Основные возможности программного комплекса

Среди основных возможностей разработанного программного комплекса можно выделить следующие.

1. Возможность декларативного определения графического интерфейса с помощью специального языка разметки XAML, что позволяет отделять функциональную логику от пользовательского интерфейса.

2. Такие механизмы, как стили и ресурсы, позволяют стандартизировать форматирование и многократно использовать его по всему приложению, а также позволяют изменить способ отображения элементов.

3. Аппаратное ускорение графики, то есть все компоненты пользовательского интерфейса визуализируются с помощью видеокарты, что повышает производительность приложения.

4. Независимость от разрешения экрана. Поскольку в WPF размеры всех элементов вычисляются в независимых от устройства единицах, приложения на WPF легко масштабируются под экраны с разным разрешением.

Платформа ASP.NET Core представляет технологию, предназначенную для создания различного рода веб-приложений [Троелсон, 2015]. ASP.NET Core построен на основе кроссплатформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS X и Linux. ASP.NET Core включает в себя фреймворк MVC, который объединяет функциональность MVC, Web API и Web Pages.

Благодаря модульности платформы, все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget.ASP.NET Core. Платформа построена из набора относительно независимых компонентов (middleware), поэтому стало возможным быстро и без особого труда расширять встроенные компоненты и с легкостью

конфигурировать канал обработки запроса. Для обработки запросов используется новый конвейер HTTP, который представляет собой улучшенную версию спецификации OWIN и Project Katana, а его модульность позволяет легко добавить дополнительные опции обработки запроса. ASP.NET Core с легкостью интегрируется с популярными фреймворками и технологиям: EntityFrameworkCore, Angular 2, RabbitMQ, SignalR 2.

Методология разработки программного обеспечения

Помимо выбранной платформы и технологии разработки, немаловажную роль играет методология разработки программного обеспечения. Разработка программного комплекса ведется в соответствии с каскадным методом, который определяет следующий жизненный цикл проекта: анализ требований, планирование, реализацию, тестирование, установку, поддержку.

Следуя парадигме каскадной модели, разработка переходит от одной стадии к другой строго последовательно. Сначала полностью завершается этап «анализа требований», в результате чего получается список требований к программному комплексу. После того как требования полностью определены, происходит переход к планированию, в ходе которого создаются документы, подробно описывающие способ и план реализации указанных требований. После того, как планирование полностью выполнено, выполняется реализация полученного проекта. На следующей стадии процесса происходит интеграция отдельных компонентов, разрабатываемых различными командами программистов. После того как реализация и интеграция завершены, производится тестирование продукта. На этой стадии устраняются все ошибки, появившиеся на предыдущих стадиях разработки. Затем программный продукт внедряется и обеспечивается его поддержка – внесение новой функциональности и устранение недочетов.

Одним из требований к программному комплексу является модульная, расширяемая архитектура [Гамма, Хелм, Джонсон, Влиссидес, 2016]. В программном комплексе каждый модуль представляет собой несколько небольших DLL библиотек, каждая из которых выполняет определенную задачу: *.Core – функционал модуля; *.Api – открытые интерфейсы для взаимодействия с другими модулями; *.Visual – графическое представление и пользовательские элементы управления, которые требуются модулю для работы; *.Integration – интеграция с основным приложением Processor.

Для управления зависимостями между модулями используется стороннее решение с открытым исходным кодом – Autofac. Таким образом, каждое клиентское приложение представляет собой контейнер модулей и его функционал можно динамически расширять или менять, присоединяя какой-либо компонент. В качестве надежного поставщика

дополнительных модулей используется серверное приложение (Server), и процесс развертывания нового модуля представлен отдельным функционалом, который доступен пользователям и не доставляет проблем при использовании.

В свою очередь, все сервисы, которыми оперирует серверное приложение, также имеют модульный вид и представляют собой динамически подключаемые библиотеки.

Используя принципы построения и компоновки модулей, а также возможности платформы WPF, становится возможным построение богатого, динамического пользовательского интерфейса, который может расширяться и допускает подключение дополнительных модулей во время работы приложения.

Модульная архитектура – ключевой залог успеха развития программного комплекса, который позволяет с легкостью добавлять новый функционал к существующим приложениям и развивать их, подстраиваясь под конкретные задачи.

Структура таблиц базы данных

Основными данными проекта являются данные о результатах какого-либо эксперимента. Каждый эксперимент, помимо экспериментальных данных, которые могут быть произвольного типа (целые, вещественные числа, вектора, массивы байтов, строковые и т.д.), и информации об объекте эксперимента, определяет свою схему (набор параметров, по которым производились измерения) и метаинформацию: время, пользователи, их права и результаты работы. В разных ситуациях организация схемы БД может отличаться от предлагаемой и, конечно же, зависит от реальных потребностей. Мы следуем фундаментальным концепциям организации БД, которые можно изучить по материалам [Дэйт, 2017; Бен-Ган, 2013; Уорсли, Дрейк, 2003]. Ниже будут кратко описаны некоторые таблицы, а таблицы, которые могут быть описаны без конкретного указания их семантики, будут опущены.

В связи с этим предлагаем следующую структуру таблиц БД.

1. *Таблица UnitType.* Содержит информацию о видах изделий.

   Список полей:

   unit_type_id. Идентификатор вида объекта.

   unit_type_name. Пользовательское имя.

   comment. Дополнительная информация.

2. *Таблица Unit.* Содержит информацию о конкретном изделии.

   Список полей:

   unit_id. Идентификатор объекта.

   unit_name. Пользовательское имя.

comment. Дополнительная информация.

unit_type_d. Идентификатор таблицы UnitType (вид объекта).

3. *Таблица Model.* Содержит информацию о модели объекта.

   Список полей:

   model_id. Идентификатор модели объекта.

   model_name. Пользовательское имя.

   unit_id. Идентификатор таблицы Unit (объект).

   comment. Дополнительная информация.

4. *Таблица Testing.* Хранит информацию об испытании, включая информацию о схеме параметров и пользователе, проводившем испытание.

   Список полей:

   testing_id. Уникальный идентификатор тестирования.

   testing_name. Наименование тестирования.

   testing_date. Дата и время тестирования.

   comment. Дополнительная информация.

   testing_type_id. Тип тестирования.

   scheme_id. Схема параметров.

   user_id. Пользователь, проводивший тестирования.

5. *Таблица Scheme.* Определяет набор параметров, по которым проводился эксперимент.

Так как данные о результатах испытаний по своей природе имеют иерархичную структуру, то для получения их денормализованного вида используется один из инструментов СУБД – представление (View), выполняющее внешнее соединение таблиц по их ключевым атрибутам.

6. *Таблица Parameter.* Определяет тип параметра, его кодовое название, также могут быть добавлены отдельной таблицей теоретические и эмпирические ограничения на параметр, рассчитанные по заданным правилам.
7. *Таблица ParameterScheme.* В связи с тем, что параметры и схемы имеют логическую связи «многие ко многим», было создано отдельное отношение, сопоставляющее параметр и схему тестирования.

   Список полей:

   parameter_scheme_id. Уникальный идентификатор

   parameter_id. Идентификатор параметра.

   scheme_id. Идентификатор схемы.

   parameter_alias. Наименование параметра в схеме.

8. *Таблица TestingData.* Содержит непосредственно сами экспериментальные данные.

Предполагается, что будут использоваться не только вещественные числа, а данные любого типа, поэтому в таблице сохраняется их байтовое представление.

Список полей:

testing_data_id. Уникальный идентификатор.

testing_id. Идентификатор испытания.

parameter_id. Идентификатор параметра.

value. Данные тестирования.

Для хранения пользовательской информации о наборах тестовых данных и графиков созданы две дополнительные таблицы – ProjectTesting и ProjectGraphics.

9. *Таблица User.* Любые действия относительно данных могут производиться только авторизованными пользователями. Данная таблица будет содержать данные о пользователях, включая логин и пароль пользователя, хранимый в закрытом виде.
10. *Таблица Role.* Определяет роль пользователей.

Список полей:

roleId. Уникальный идентификатор.

role_name. Уникальный код роли

display_name. Наименование роли

11. *Таблица UserRole.* Определяет соотношение между ролями и пользователями. У каждого пользователя может быть несколько ролей. Фактически данная таблица является вспомогательной для организации связи «многие ко многим».
12. *Таблица Permission.* Определяет права доступа для пользователей и их роли.

Список полей:

permission_id. Идентификатор роли.

permission_name. Наименование роли

description. Дополнительная информация

13. *Таблица Prohibition.* Хранит коллекцию запретов на доступ к определенным объектам в БД (испытания, графики, проекты, параметры). Одна строка соответствует запрету для пользователя или роли.

Список полей:

prohibition_id. Уникальный идентификатор.

role_id. Уникальный идентификатор роли

user_id. Уникальный идентификатор пользователя.

permission_id Уникальный идентификатор прав.

object_id. Идентификатор объекта БД.

object_type. Тип объекта БД.

14. *Таблица Graphics.* Определяет визуальные данные (график), построенный на тестовых данных. Сама таблица содержит только метаинформацию, включая время и пользователя, создавшего график, описание и т.д.
15. *Таблица GraphicsVersion.* Обеспечивает поддержку версионирования графиков и сохранение информации о пользователе, совершившим изменения.
16. *Таблица GraphicsData.* Определяет непосредственно сам график, построенный пользователем системы.

Модули программного комплекса

Модуль администрирования

Данный модуль предназначен для управления списком пользователей, зарегистрированных в системе, а также для управления ролями пользователей.

Структура визуального представления модуля выглядит следующим образом:

1. Документ управления пользователями:

– просмотр списка всех пользователей с возможностью поиска по имени;

– регистрация нового пользователя;

– редактирование информации о зарегистрированном пользователе;

– включение / выключение пользователя;

– назначение пользователю ролей;

– исключение пользователя из роли;

2. Документ управления ролями:

– просмотр списка всех ролей в системе с возможностью поиска по имени;

– создание новой роли;

– редактирование информации о существующей роли;

– просмотр списка всех пользователей, которым назначена определенная роль.

Для взаимодействия с сервером в документе пользователей используются следующие модели:

1. UserInfo – представляет собой всю базовую информацию о пользователях.
2. UserDetail – содержит список ролей пользователя. Расширяет UserInfo.
3. UserRegisterInfo – содержит информацию для регистрации нового пользователя. Расширяет UserInfo.

Для взаимодействия с сервером в документе ролей используются такие модели:

1. RoleInfo – содержит базовую информацию о роли.

2. RoleDetail – содержит список пользователей, принадлежащих этой роли. Расширяет RoleInfo.

Модуль импорта данных

Импортирование данных происходит через отдельное приложение, которое решает следующие задачи:

– загрузка данных в формате MS Excel или MS Word;

– выделение схемы сопоставления параметров или загрузка существующей схемы сопоставления;

– выделение параметров по схеме;

– отправка данных на сервер

– проверка корректности данных;

– запись в базу данных.

– запись информации о пользователе, внесшим изменения в систему, для дополнительного контроля.

Структура визуального представления модуля выглядит следующим образом:

– документ загрузки данных об эксперименте;

– загрузка данных в формате MS Excel, MS Word;

– выбор схемы сопоставления;

– выделение параметров испытания по схеме сопоставления;

– документ выделения схемы сопоставления данных;

– автоматическая генерация схемы сопоставления;

– редактирование сгенерированной схемы пользователем;

– документ просмотра списка испытаний;

– поиск испытаний по заданному фильтру;

– отображение эксперимента в иерархии.

Для взаимодействия с сервером в документе создания и редактирования объектов используются следующие модели:

1. EntityInfoBase. Содержит базовую информацию об объекте
2. EntityInfo. Содержит идентификатор объекта, сохраненного в базе данных.
3. EntityCreateInfo. Служит для создания нового объекта. Содержит идентификатор родительского объекта в иерархии. Расширяет EntityInfoBase.

Модуль экспорта данных

Данный модуль предназначен для вывода в разных форматах результатов эксперимента. Например, экспорт построенных по экспериментальным данным графиков в форматы MS Word,

MS Excel, PDF

Также предусмотрена возможность расширения списка поддерживаемых форматов (например, формат html для отображения графиков через web-интерфейс) и реализация выгрузки данных в формат другого специализированного программного обеспечения для дальнейшей обработки.

Графоаналитический модуль

Данный модуль предназначен для графического представления и визуального анализа, исследования данных, графического редактирования копии результатов эксперимента, математической обработки.

Graphing – компонент, содержащий алгоритмы отображения данных, а также реализующий функционал для работы с графиками. Основная идея заключается в возможности быстрой отрисовки данных, а также редактирования точек и самих графиков.

В данном компоненте необходимо предусмотреть функционал начальной настройки для последующей отрисовки как самого графического отображения данных, так и графических атрибутов этого отображения (масштаб, координатная сетка); отображение набора графиков с индивидуальными настройками для каждого графика. В данном случае реализация этого компонента предлагает определенный набор стандартных решений типичных задач с возможностью индивидуальной настройки состава набора и параметров отображения, то есть при графическом отображении часто возникает типичный набор задач, которые необходимо решить. Причем решение этих задачи стандартно и может быть найдено в открытых источниках.

Модуль совместной работы

Модуль предназначен для организации совместной работы операторов, соединенных между собой локальной сетью. То есть программный комплекс предоставляет возможность передавать изменения одного оператора другому посредством сообщений. Для выполнения подобных задач существует решение с открытым исходным кодом – очередь сообщений RabbitMQ, общая логика работы с которой описана ниже.

Каждое клиентское приложение (Processor) при совершении каких-либо действий над проектом посылает соответствующее сообщение серверу. Тот, в свою очередь, перенаправляет его системе очередей RabbitMQ, которая распределяет его всем остальным получателям, то есть другим пользователям, у которых открыт текущий проект. Таким образом, все пользователи, работающие над одним проектом, могут видеть общие изменения в режиме реального времени. Данный механизм позволяет пользователям совместно работать над одним проектом и координировать общие результаты.

Развертывание программного комплекса осуществляется в три этапа:

1. Подготовка сервера баз данных SQL.

2. Подготовка и развертывание Web-приложения Server.

3. Установка клиентских приложений.

В качестве сервера баз данных подойдет любая машина под управлением операционной системы Linux / Windows.

WServer – web-приложение, поэтому для его развертывания требуется web-сервер. В зависимости от предпочтений, следует использовать web-сервер IIS 8.0 под управлением операционной системы MicrosoftServer 2012 и выше или же сервер с открытым исходным кодом Kestrel под управлением любого дистрибутива Linux.

Среди требований к клиентским компьютерам можно назвать следующие:

1.) операционная система Windows XP и выше;

2) платформа MS .NET Framework версии не ниже 4.6.2.

Заключение

В статье продемонстрирована стратегия разработки программного комплекса для обработки данных. Предложена методика и упрощенный вариант схемы базы данных для хранения данных эксперимента. Перечислены необходимые модули приложения и средства их реализации.

Также указаны существенные аспекты, на которые нужно обращать внимание при разработке такого рода программных продуктов. Стоит иметь в виду, что вынесение бизнес-логики за пределы клиентских приложений позволяет существенно снизить требования к рабочим компьютерам операторов, которые используется для обработки экспериментальных данных, что позволяет снизить расходы в рабочем процессе. Разработанная архитектура программного комплекса может быть применена практически во всех отраслях экономики для решения экономико-статистических, научно-технических и организационно-технологических задач, предусматривающих построение феноменологичсеких математических моделей и/или сопоставление экспериментальных данных с результатами математического моделирования.

Библиография

1. Бен-Ган И. Microsoft SQL Server 2012. Высокопроизводительный код T-SQL. Оконные функции. М.: Русская Редакция; СПб.: БХВ-Петербург, 2013.
2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2016.
3. Дэйт К.Дж. Введение в системы баз данных. М.: Вилльямс, 2017.

4. Мак-Дональд М. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов. М.: Вильямс, 2015.
5. Поляков В.П. Развитие информационной подготовки в контексте Стратегии национальной безопасности Российской Федерации // Наукоград: наука, производство, общество. 2016. № 2. С. 46-51.
6. Троелсон Э. Язык программирования C# 5.0 и платформа .NET 4.5. М.: Вильямс, 2015.
7. Уорсли Дж., Дрейк Дж. PostgreSQL. Для профессионалов. СПб.: Питер, 2003.

Adaptive architecture of software and hardware complex for data storage and processing

Yurii A. Kostikov
PhD in Physical and Mathematical Sciences,
Head of the Department 812,
Moscow Aviation Institute (National Research University),
125993, 4 Volokolamskoe shosse, Moscow, Russian Federation;
e-mail: jkostikov@mail.ru

Vitalii Yu. Pavlov
PhD in Physical and Mathematical Sciences,
Head of the Department 318,
Moscow Aviation Institute (National Research University),
125993, 4 Volokolamskoe shosse, Moscow, Russian Federation;
e-mail: kafedra@istmati.ru

Aleksandr M. Romanenkov
PhD in Technical Sciences, Associate Professor,
Department 812,
Moscow Aviation Institute (National Research University),
125993, 4 Volokolamskoe shosse, Moscow, Russian Federation;
e-mail: romanaleks@gmail.com

Vladimir B. Ternovskov

PhD in Technical Sciences, Associate Professor,

Department 318,

Moscow Aviation Institute (National Research University),

125993, 4 Volokolamskoe shosse, Moscow, Russian Federation;

e-mail: vternik@mail.ru

Abstract

This article describes the methodology of development of software complex of data storage, processing and preparation for constructing mathematical models and comparison of results of mathematical modeling with the results of tests and experiments. The software complex is classic multiuser information system, realized with the use of modern computer technology. This software complex has a flexible modular architecture, including data bank and various interfaces for interacting with these data and their conversion into required formats. The structure and algorithms of interaction of modules can vary depending on the types of data and tasks. The authors of this article consider some important aspects to which we should pay attention when developing such software products. It is worth to note that the imposition of business logic outside of the client applications can significantly reduce the requirements for operator's computer, which is used for experimental data processing, that allows you to reduce costs in the workflow. The authors notice that architecture of the software considered in this article can be applied in almost all sectors of the economy to solve the economic-statistical, scientific-technical and technological problems, including the construction of a phenomenological mathematical models and/or comparison of the experimental data with the results of mathematical modeling.

For citation

Kostikov Yu.A., Pavlov V.Yu., Romanenkov A.M., Ternovskov V.B. (2017) Adaptivnaya arkhitektura programmno-apparatnogo kompleksa khraneniya i obrabotki dannykh [Adaptive architecture of software and hardware complex for data storage and processing]. *Ekonomika: vchera, segodnya, zavtra* [Economics: Yesterday, Today and Tomorrow], 7(9A), pp. 192-207.

Keywords

Software complex, data storage, data processing, mathematical model, interface, database, WPF, ASPCore .netframework 4.6.

References

1. Ben-Gan I. (2013) Microsoft SQL Server 2012. *Vysokoproizvoditel'nyi kod T-SQL. Okonnye funktsii* [Microsoft SQL Server 2012. High-performance T-SQL code. The window function]. Moscow: Russkaya Redaktsiya Publ.; Saint Peter: BKhV-Peterburg Publ.
2. Deit K.Dzh. (2017) *Vvedenie v sistemy baz dannykh* [Introduction to database systems]. Moscow: Vill'yams Publ.
3. Gamma E., Khelm R., Dzhonson R., Vlissides Dzh. (2016) Priemy ob''ektno-orientirovannogo proektirovaniya. Patterny proektirovaniya [Techniques of object-oriented design. Design patterns]. Saint Petersburg: Piter Publ.
4. Mak-Donal'd M. (2015) *WPF: Windows Presentation Foundation v .NET 4.5 s primerami na C# 5.0 dlya professionalov* [Windows Presentation Foundation in .NET 4.5, with examples in C# 5.0 for professionals]. Moscow: Vill'yams Publ.
5. Polyakov V.P. (2016) Razvitie informatsionnoi podgotovki v kontekste Stra-tegii natsional'noi bezopasnosti Rossiiskoi Federatsii [Development of information training in the context of the National Security Strategy of the Russian Federation]. *Naukograd: nauka, proizvodstvo, obshchestvo* [Naukograd: science, science, production and society], 2, pp. 46-51.
6. Troelson E. (2015) *Yazyk programmirovaniya C# 5.0 i platforma .NET 4.5* [The programming language C# 5.0 and the platform .NET 4.5]. Moscow: Vill'yams Publ.
7. Uorsli Dzh., Dreik Dzh. (2003) *PostgreSQL. Dlya professionalov* [PostgreSQL. For professionals]. Saint Petersburg: Piter Publ.