ЕУРАЗИЯ ҰЛТТЫҚ УНИВЕРСИТЕТІ

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
ТҰҢҒЫШ ПРЕЗИДЕНТІ - ЕЛБАСЫНЫҢ ҚОРЫ

«ҒЫЛЫМ ЖӘНЕ БІЛІМ – 2017»

студенттер мен жас ғалымдардың
XII Халықаралық ғылыми конференциясының
БАЯНДАМАЛАР ЖИНАҒЫ

СБОРНИК МАТЕРИАЛОВ
XII Международной научной конференции
студентов и молодых ученых
«НАУКА И ОБРАЗОВАНИЕ – 2017»

PROCEEDINGS
of the XII International Scientific Conference
for students and young scholars
«SCIENCE AND EDUCATION - 2017»

14th April 2017, Astana

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
Л.Н. ГУМИЛЕВ АТЫНДАҒЫ ЕУРАЗИЯ ҰЛТТЫҚ УНИВЕРСИТЕТІ

«Ғылым және білім - 2017»
студенттер мен жас ғалымдардың
XII Халықаралық ғылыми конференциясының
БАЯНДАМАЛАР ЖИНАҒЫ

СБОРНИК МАТЕРИАЛОВ
XII Международной научной конференции
студентов и молодых ученых
«Наука и образование - 2017»

PROCEEDINGS
of the XII International Scientific Conference
for students and young scholars
«Science and education - 2017»

2017 жыл 14 сәуір

Астана

УДК 378

ББК 74.58
Ғ 96

Ғ 96
«Ғылым және білім – 2017» студенттер мен жас ғалымдардың XII Халықаралық ғылыми конференциясы = The XII International Scientific Conference for students and young scholars «Science and education - 2017» = XII Международная научная конференция студентов и молодых ученых «Наука и образование - 2017». – Астана: http://www.enu.kz/ru/nauka/nauka-i-obrazovanie/, 2017. – 7466 стр. (қазақша, орысша, ағылшынша).

ISBN 978-9965-31-827-6

Жинаққа студенттердің, магистранттардың, докоранттардың және жас ғалымдардың жаратылыстану-техникалық және гуманитарлық ғылымдардың өзекті мәселелері бойынша баяндамалары енгізілген.

The proceedings are the papers of students, undergraduates, doctoral students and young researchers on topical issues of natural and technical sciences and humanities.

В сборник вошли доклады студентов, магистрантов, докторантов и молодых ученых по актуальным вопросам естественно-технических и гуманитарных наук.

УДК 378

ББК 74.58

ISBN 978-9965-31-827-6

$$\frac{1}{3}\left[\left\|\sigma^{m+1/3}\right\|_{B_1}^2+\left\|\vartheta^{m+1/3}\right\|^2\right]_{\bar{t}}+\left\|\sigma^{m+1/3}\right\|_{C_1}^2=\frac{1}{3}\left[\left\|\sigma^{m}\right\|_{B_1}^2+\left\|\vartheta^{m}\right\|^2\right]_{\bar{t}}+\left\|\sigma^{m}\right\|_{C_1}^2,$$

(16)

а для (14), (15)

$$\frac{1}{3}\left[\left\|\sigma^{m+2/3}\right\|_{B_1}^2+\left\|\vartheta^{m+2/3}\right\|^2\right]_{\bar{t}}+\left\|\sigma^{m+2/3}\right\|_{C_1}^2=\frac{1}{3}\left[\left\|\sigma^{m+1/3}\right\|_{B_1}^2+\left\|\vartheta^{m+1/3}\right\|^2\right]_{\bar{t}}+\left\|\sigma^{m+1/3}\right\|_{C_1}^2, \quad (17)$$

$$\frac{1}{3}\left[\left\|\sigma^{m+1}\right\|_{B_1}^2+\left\|\vartheta^{m+1}\right\|^2\right]_{\bar{t}}+\left\|\sigma^{m+1}\right\|_{C_1}^2=\frac{1}{3}\left[\left\|\sigma^{m+2/3}\right\|_{B_1}^2+\left\|\vartheta^{m+2/3}\right\|^2\right]_{\bar{t}}+\left\|\sigma^{m+2/3}\right\|_{C_1}^2, \quad (18)$$

Теперь из (16)-(18) для аддитивной разностной схемы(13), (14), (15) вытекает справедливость сеточного закона сохранения (10)

$$J^{m+1}=J^m=\ldots=J^0. \quad (19)$$

Остается заметить, что при дальнейших стандартных рассуждениях, приводящих от (19) к теореме сходимости, следует рассматривать аппроксимацию задачи (1) – (7) разностной схемой (13) - (15) как суммарную [8, гл. IX, § 3].

Список использованных источников

1. Букенов М. М. Постановка динамической задачи линейной вязкоупругости в скоростях напряжений, Сиб. журн. вычисл. матем., – 2005, Т. 8, № 4. – С. 289–295.
2. Попов Ю.П., Самарский А.А. Полностью консервативные разностные схемы // Журн. вычисл. матем. и мат. физики. – 1969. – Т. 9, № 4. – С. 953–958.
3. Самарский А.А., Попов Ю.П. Разностные схемы газовой динамики. – М.: Наука, 1975.
4. Багриновский К.А., Годунов С.К. Разностные схемы для многомерных задач // Докл. АН СССР. – 1957. – Т. 115, № 3. – С. 431–433.
5. Анучина Н.Н., Яненко Н.Н. Неявные схемы расщепления для гиперболических уравнений и систем // Докл. АН СССР. – 1959. – Т. 128, № 6. – С. 1103–1106.
6. Яненко Н.Н. Об экономичных неявных схемах (метод дробных шагов) // Докл. АН СССР. – 1960. – Т. 134, № 5. – С. 1034–1036.
7. Кристенсен Р. Введение в теорию вязкоупругости. – М.: Мир, 1974.
8. Самарский А.А. Теория разностных схем. – М.: Наука, 1983.

УДК 681.51

ИНТЕГРАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ С ДАННЫМИ РЕЛЯЦИОННОЙ СУБД

Альбеков Бахтияр Кайратович
albekovbk@gmail.com
Боромбаева Акмарал Бакыткызы
borombayeva@gmail.com
Студенты кафедры Математического и компьютерного моделирования
ЕНУ им. Л.Н.Гумилева, Астана, Казахстан
Научные руководители - Муканова Б.Г., Ракишева Д.С.

В данный момент в «АО «НК Казахстан Темир Жолы» на базе Главного вычислительного центра ведется разработка автоматизированной системы «Единый корпоративный центр начислений и расчетов – ЕКЦР». Система создается как информационный ресурс, обеспечивающий всем заинтересованным сторонам централизованный доступ к

показателям объемов услуг, оказываемых участниками перевозочного процесса. Основной целью системы является оптимизация взаиморасчетов между компаниями-участниками. Архитектура системы представлена ниже на Схеме №1.

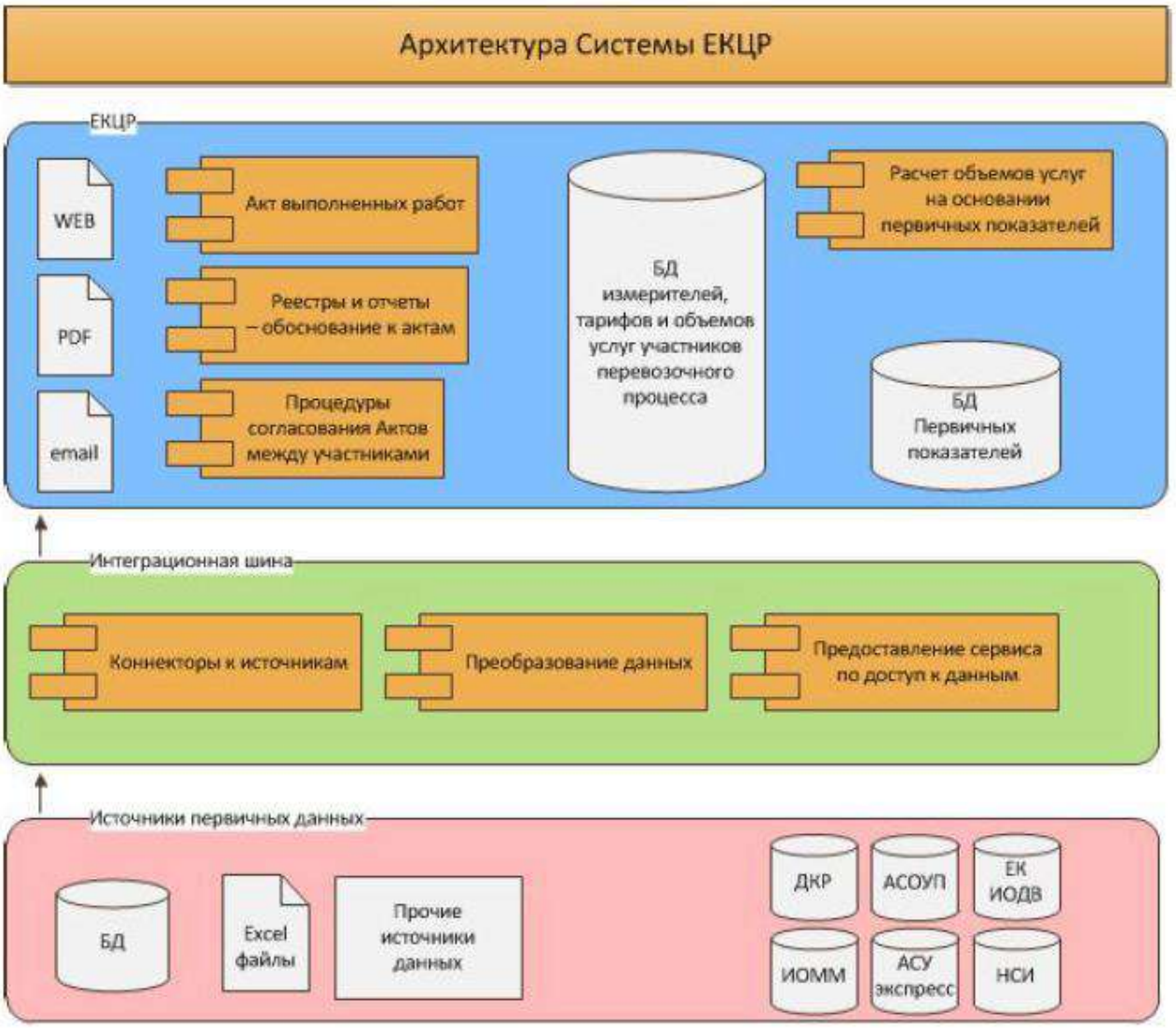

Схема №1. Архитектура АС ЕКЦР

Источники первичных данных – различные информационные системы, сервисы, базы данных и прочие источники информации, в форме файлов.

Интеграционная шина – сервисная шина предприятия (англ. Enterpriseservicebus, ESB) – связующее программное обеспечение, обеспечивающее централизованный и унифицированный событийно-ориентированный обмен сообщениями между различными информационными системами на принципах сервис-ориентированной архитектуры. Задачей шины является предоставление доступа ЕКЦР к данным, необходимым для расчета той или иной услуги.

ЕКЦР – приложение, предоставляющее пользователями сервисы по расчету, хранению и предоставлению доступа посредством WEB-интерфейса к данным отражающим объем оказываемых услуг, а также хранения необходимых первичных данных. Приложение реализуется на принципах клиент-серверной архитектуры.

В зависимости от особенностей определенного бизнес-процесса пользователь приложения получает доступ к определенным данным. Далее рассмотрим простой пример реализации одного из таких моментов.

Допустим, что у нас есть договор между неким заказчиком и АО «Военизированная железнодорожная охрана» о закупке услуг по охране имущества юридических и физических лиц, в том числе при его транспортировке. Как и в любом бизнес-процессе для корректной оплаты оказанных услуг Исполнитель должен предоставить Заказчику Акт выполненных работ (оказанных услуг). Однако, акт содержит лишь итоговые данные и суммы, поэтому вполне очевидно, что обе стороны заинтересованы в подробных отчетах, подтверждающих прозрачность процесса. И так, конкретная задача на данном примере будет сформулирована следующим образом:

- Необходимо создать справку (отчет) за месяц о количестве вагонов (всех собственников), завершивших перевозку.

Коротко о применяемом программном обеспечении для решения данной задачи:

1) IntelliJIDEA – интегрированная среда разработки, в ее коммерческой версии реализована полная поддержка нужных нам JavaEE, Groovy, SQL и т.д. Весь проект размещен в данной IDE. К тому же она позволяет работать с консолью базы данных без применения сторонних программ.
2) JavaEE – JavaEnterpriseEdition, представляет собой набор спецификаций для созданий программного обеспечения уровня предприятия.
3) GroovyServerPages – используется для создания веб-приложений, дает возможность смешивать статический и динамический контент на одной странице.
4) MySQL – свободная реляционная система управления базами данных.
5) ApacheTomcat – контейнер сервлетов, программа представляющая собой сервер.

Непосредственное создание отчета начинается с того, что в соответствующем файле проекта “ekcr” необходимо внести запись о создаваемом отчете. Запись имеет вид:

<iid=”21” nm=”Test3 report” gsp=”Test3.html”/> ,

где id – номер отчета, nm – его имя, которое будет отображаться на странице, igsp – имя gsp-файла, в котором будет написан gsp-код этого отчета.

Далее создаемhtml-шаблон таблицы, которая будет содержать в себе данные отчета. Также нужно понимать, что данные будут подгружаться в эту таблицу динамически, то есть они не будут заданы в коде таблицы изначально, а будут получены из базы данных. Это говорит нам о том, что необходимо создать .gsp файл. Фактически GSP-страница является просто стандартным HTML с добавлением тегов Grails для представления динамического контента. Сейчас достаточно создать шапку (рис.2) необходимой нам таблицы с помощью простыхhtml-тэгов типа <table>, <tr>, <th>. В данном случае мы должны создать таблицу как на рис.1.

Справка

о количестве вагонов (всех собственников), завершивших перевозку

Дата	кол-во вагонов проследовавших по НОД за сутки		кол-во вагонов завершивших рейс за сутки		кол-во вагоно/часов		вагоно/часы с охраняемым грузом по отделениям перевозок					кол-во записей в базе	размеры базы (Кб)	размер файла (Кб)
	всего	в т.ч. с охраняемым грузом	всего	в т.ч. с охраняемым грузом	всего	в т.ч. с охраняемым грузом	НОД-01	НОД-02	...	НОД-13	НОД-14			
01.01.2016														
02.01.2016														
...														
30.01.2016														
31.01.2016														
итого														

рис.1. Шаблон таблицы

```
<table class="ekcr-report-table" style="table-layout:fixed; text-align:center; vertical-align:middle"; cellpadding=5;>
    <caption><h2 align="center">Справка за ${DateTimeFormat.forPattern("MMM").print(params.getValueDateTime("dt1"))}. месяц<br/>
        о количестве вагонов (всех собственников), завершивших перевозку грузов</h2></caption>
    <tr>
        <th colspan="1" rowspan="2">Дата</th>
        <th colspan="2" rowspan="1">Кол-во вагонов<br/> проследовавших по <br/> НОД за сутки</th>
        <th colspan="2" rowspan="1">Кол-во вагонов<br/> завершивших рейс <br/> за сутки</th>
        <th colspan="2" rowspan="1">Кол-во вагоно/часов</th>
        <th colspan="14" rowspan="1">Вагоно/часы с охраняемым грузом по отделениям перевозок</th>
    </tr>

    <tr>
        <th colspan="1" rowspan="1">Всего</th>
        <th colspan="1" rowspan="1">В т.ч. с <br/> охраня- <br/>емым <br/>грузом</th>
        <th colspan="1" rowspan="1">Всего</th>
        <th colspan="1" rowspan="1">В т.ч. с <br/> охраня- <br/>емым <br/> грузом</th>
        <th colspan="1" rowspan="1">Всего</th>
        <th colspan="1" rowspan="1">В т.ч. с <br/> охраня- <br/>емым <br/> грузом</th>
        <% for(def rec3 : store3) { %>
        <th colspan="1" rowspan="1">НОД-${rec3.getValue("nod")}</th>
        <% } %>
    </tr>
```

рис.2. html-код шапки таблицы

Теперь необходимо создать Groovy-class, с помощью которого мы будем извлекать информацию из базы данных.

```
package ekcr.main.model.dao.report

import ...

class Test3 extends WaxListDao {

    @DaoMethod
    public DataBox load(IVariantMap params) {
        def Box = new DataBox()
            def store = ut.loadSql("""...""",
                    [dt1: params.getValueDateTime("dt1")])

            def store2 = ut.loadSql("""...""")

            def store3 = ut.loadSql("""...""")

            Box.store = store;
            Box.store2 = store2;
            Box.store3 = store3;
        return Box
    }
}
```

рис.3. Шаблон Groovy-класса

Задачей этого класса является сохранение извлеченных данных в специальные хранилища «store». Только после того как данные появятся в «сторах», можно будет отобразить их в таблице веб-интерфейса. Также в стор загружаются и параметры, которые указывает пользователь в веб-интерфейсе. Так как store-хранилища содержат данные лишь одного запроса, а в нашем случае, вывести отчет в одном запросе невозможно, то мы создаем несколько store и помещаем их в единый DataBox.

В аргументе функции ut.loadSqlв кавычках пишется SQL-запрос. Написание запросов является едва ли не ключевой частью создания отчетов, так как оказывает наибольшее влияние на быстродействие WEB-интерфейса, который будет выводить в отчет огромное количество данных.

```
select tripbynod.nod,
  date(trip.dt) dt,
  date_format(dt, '%d.%m.%Y') dt11,
  count(*) total, CAST(sum(guard=1) AS UNSIGNED) totalguard,
  CAST(sum(CASE WHEN trip.destinationStation=tripbynod.endStation THEN 1 ELSE 0 END) AS UNSIGNED) totalend,
  CAST(sum(CASE WHEN trip.destinationStation=tripbynod.endStation THEN guard=1 ELSE 0 END) AS UNSIGNED) totalendguard,
  CAST(sum(timestampdiff(hour, tripbynod.startTime, tripbynod.endTime)) AS UNSIGNED) totalwh,
  CAST(sum(CASE WHEN guard=1 THEN timestampdiff(hour, tripbynod.startTime, tripbynod.endTime) ELSE 0 END) AS UNSIGNED) totalwhguard
from trip join tripbynod on trip.id=tripbynod.trip
where MONTH(dt)=MONTH(:dt1) AND YEAR(dt) = YEAR(:dt1) AND nod<>0 AND nod<=14 group by date(dt);
```

рис.4. Запрос-1

На рисунке №4 дан запрос, который был написан для вывода столбца дат в определенном виде, и 6 следующих столбцов, данные в которых соответствуют дате из первой колонки. Эти данные можно было получить объединением нескольких запросов, но это было бы крайне неэффективно. Столбцы содержат очень похожие данные, то есть мы понимаем, что это данные одной таблицы, однако они получены путем применения абсолютно разных и несовместимых условий. Все же это не повод, чтобы выводить эту пару таблиц каждый раз с новым условием, жертвуя при этом производительностью системы.

```
select dt, nod,
  CAST(sum(CASE WHEN guard=1 THEN timestampdiff(hour, tripbynod.startTime, tripbynod.endTime) ELSE 0 END) AS UNSIGNED) whbynod
from trip join tripbynod on trip.id=tripbynod.trip
where nod<>0 AND nod<=14
group by dt, nod;
```

рис.5. Запрос-2

Запрос-2 передает в store2 количество вагоно/часов сгруппированных по дате и НОДам для вагонов с охраняемым грузом. Если обратиться к форме таблицы, то это данные, которые будут размещены в ячейках под НОДами в соответствии с датой. В результате sql-запроса эти данные отображаются в виде 3 столбцов: дата, НОД, вагоно/часы. Поэтому позже предстоит поработать над корректным отображением этих данных в html-шаблоне.

```
select distinct nod,
  CAST(sum(CASE WHEN guard=1 THEN timestampdiff(hour, tripbynod.startTime, tripbynod.endTime) ELSE 0 END) AS UNSIGNED) whbynod
from tripbynod join trip on trip.id=tripbynod.trip
where nod<>0 and nod<=14
group by nod;
```

рис.6. Запрос-3

Последний запрос извлекает список НОДов и соответствующее им количество вагоно/часов для вагонов с охраняемым грузом. Список НОДов нам нужен для отображения этого списка в шапке таблицы, тогда она будет динамична и не вызовет критических ошибок, в случае, реорганизации НОДов. Вагоно/часы выводимые в этом запросе, являются суммой за месяц, то есть будут выведены в последней строке таблицы.

Остается сформировать gsp-файл. После импорта необходимых файлов, обеспечиваем чтение параметров и инициализируем «бокс»-данных с его «сторами»-данных.

```
<%
    WaxTml tml = new WaxTml(this)
    def params = tml.params


    def report21Dao = tml.createDao(Test3.class)
    jandcode.dbm.data.DataBox Box = report21Dao.load(params)
    DataStore store = Box.store
    DataStore store2 = Box.store2
    DataStore store3 = Box.store3


    def calcSum = {st, fieldName ->
        def s = 0
        for (DataRecord rec : st){
            s += rec.getValue(fieldName)
        }
        return s
    }

%>
```

рис.7. DataBoxи функцияCalcSum в gsp-файле

Также создаем функцию, которая будет вычислять сумму задаваемых нами field-ов (столбцов) таблицы. Она нужна нам, чтобы выводить сумму в итоговую строку.

Так как шапка таблицы уже создана *(рис. 2.)*, то остается лишь уточнить моменты касательно динамеской части кода шапки. Там где должны создаваться ячейки с НОДами, мы создаем цикл, который извлекает из store3 список НОДов, чтобы количество ячеек было соответствующим.

Далее пишем код для основной динамической части таблицы. Он будет состоять из двух частей: первая часть будет состоять из двух циклов, формирующих необходимое количество ячеек и заполняя их соответствующими данными; вторая часть –формирует итоговую строку с выводом суммы соответствующих колонок посредством написанной ранее функции calcSum и данныхиз store3.

```
<% for (def rec : store) { %>
<tr>
    <td>${rec.getValue("dt11")}</td>
    <td>${rec.getValue("total")}</td>
    <td>${rec.getValue("totalguard")}</td>
    <td>${rec.getValue("totalend")}</td>
    <td>${rec.getValue("totalendguard")}</td>
    <td>${rec.getValue("totalwh")}</td>
    <td>${rec.getValue("totalwhguard")}</td>
<% for (def rec2 : store2) { %>
<% if (rec.getValue("dt").equals(rec2.getValueDateTime("dt"))){ %>
    <td>${rec2.getValue("whbynod")}</td>
<% } %>
<% } %>
</tr>
<%}%>
```

рис.8. Код динамической части-1

```
    <tr>
    <td>Итого</td>
    <td>${calcSum(store, "total")}</td>
    <td>${calcSum(store, "totalguard")}</td>
    <td>${calcSum(store, "totalend")}</td>
    <td>${calcSum(store, "totalendguard")}</td>
    <td>${calcSum(store, "totalwh")}</td>
    <td>${calcSum(store, "totalwhguard")}</td>
    <% for (def rec3 : store3) { %>
    <td>${rec3.getValue("whbynod")}</td>
    <% } %>

</table>
```

рис.9. Код динамической части-2

В конечном итоге, перезагрузив наштomcat-сервер, получается справка следующего вида:

Справка за дек. месяц
о количестве вагонов (всех собственников), завершивших перевозку грузов

Дата	Кол-во вагонов проследовавших по НОД за сутки		Кол-во вагонов завершивших рейс за сутки		Кол-во вагоно/часов		Вагоно/часы с охраняемым грузом по отделениям перевозок													
	Всего	В т.ч. с охраня-емым грузом	Всего	В т.ч. с охраня-емым грузом	Всего	В т.ч. с охраня-емым грузом	НОД-1	НОД-2	НОД-3	НОД-4	НОД-5	НОД-6	НОД-7	НОД-8	НОД-9	НОД-10	НОД-11	НОД-12	НОД-13	НОД-14
01.12.2016	51462	24534	20446	7187	642980	405795	35205	31260	29932	42098	7882	22577	50073	44669	38813	22238	37091	9273	19013	15671
02.12.2016	52359	24902	14474	7242	642533	441139	32277	39317	28236	44287	4892	18448	61377	57928	55413	22838	37950	6489	18626	13061
03.12.2016	52604	18275	10558	5705	643895	370296	27112	22954	27819	38198	4707	24436	63697	41532	46936	18157	30580	3641	9931	10596
Итого	156425	67711	45478	20134	1929408	1217230	94594	93531	85987	124583	17481	65461	175147	144129	141162	63233	105621	19403	47570	39328

рис.10. Результат

Данная задача, а точнее решение имеет широкое применение не только в разработке системы «АС ЕКЦР». С развитием технологий, все больше компаний отказываются от бумажной отчетности. Государственные структуры, национальные компании, банки второго уровня и прочие крупные предприятия стремятся максимально автоматизировать любой бизнес-процесс. Разработка новых систем, а также оптимизация действующих, заняла прочное место среди ремесел нового времени.

Список использованных источников

1. Техническое задание 044525555.10701.002.ТЗ Автоматизированная система «Единого корпоративного центра начислений и расчётов» (АС ЕКЦР). С. 4-7

УДК 519.63

МОДЕЛЬ ЭЛЕКТРИЧЕСКОЙ АКТИВНОСТИ СЕРДЦА

Алмат Айжан
aizhanka.almatova@mail.ru
Студентка 4-го курса ЕНУ им. Л. Н. Гумилева
Научный руководитель – Жусупова Д. С.

В статье рассматривается математическая модель bidomain электрической активности сердца, учитывающая различные электропроводности внутриклеточного и внеклеточного пространств. Модель bidomain представляет собой набор математических уравнений, которые определяют электрические свойства сердечной ткани. Она была разработана в конце 1970 – х годов и в настоящее время широко используется при численном моделировании электрического поведения сердца.

Модель bidomain представляет собой двух – или трехмерную модель кабеля (см. Рис. 1). Это модель континуума, в том смысле, что она предсказывает электрическое поведение, усредненное по многим ячейкам. Модель учитывает различные электропроводности внутриклеточного и внеклеточного пространств.

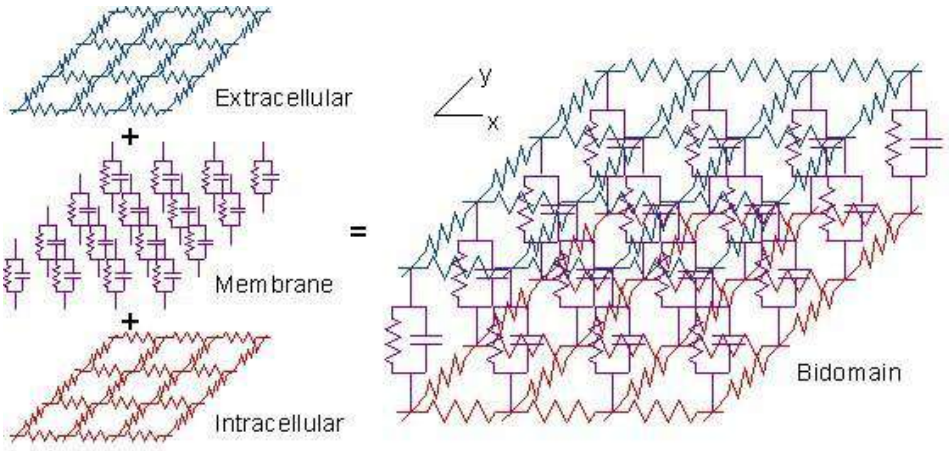

Рис. 1: Электрическая схема, которая аппроксимирует двумерную модель bidomain. Внутриклеточные и внеклеточные пространства представлены резисторными сетками. Резисторы больше в направлении, перпендикулярном волокнам (y), чем в направлении, параллельном им (x). Эти два пространства связаны клеточной мембраной, представленной в виде параллельной комбинации резистора и конденсатора Roth [1].