МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

01.03.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

(код и наименование направления подготовки)

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

(направленность (профиль))

БАКАЛАВРСКАЯ РАБОТА

на тему «**Разработка автоматизированной системы управления рабочим пространством Тольяттинского офиса компании NETCRACKER»**

Студент М.Д. Кудряшов _____

Руководитель А.И. Туищев _____

Консультант по аннотации Н.В. Ященко _____

Допустить к защите

Заведующий кафедрой к. тех. н., доцент, А. В. Очеповский_____

«_____» _____ 20___ г.

Тольятти 2017

АННОТАЦИЯ

Темой данной выпускной квалификационной работы является «Разработка автоматизированной системы управления рабочим пространством тольяттинского офиса компании «Netcracker».

Работа выполнена студентом Тольяттинского Государственного Университета, института математики, физики и информационных технологий, группы ПМИб-1301 Кудряшовым Максимом Дмитриевичем.

Актуальность темы «Разработка автоматизированной системы управления рабочим пространством тольяттинского офиса компании «NetCracker» обуславливается потребностью в разработке АСУ, которая даст возможность автоматизировать деятельность сотрудников отдела кадров и администрации офиса, а именно процесс учета рабочего пространства.

Объект выпускной квалификационной работы – процесс управления рабочим пространством.

Субъект выпускной квалификационной работы – автоматизированная система управления.

Во введении описывается актуальность проводимого исследования, выделяются проблемы исследования, формируется цель и ставятся задачи.

В первой главе проводится анализ деятельности отдела кадров и администрации офиса. Процесс учета рабочего пространства разбивается на этапы и по ним строится контекстная диаграмма всего процесса. Выделяются самые слабые этапы, нуждающиеся в автоматизации. Описываются основные принципы динамического программирования. На его основе ставится задача оптимального распределения сотрудников по офису и описывается её математическая модель.

Во второй главе описывается обоснование архитектуры проектируемой автоматизированной системы управления рабочим пространством. Описываются требования к системе по классификации требований FURPS+. Проводится обоснование выбора технологий для разработки и проектируется система. Представляются блок-схемы с изображением работы некоторых

методов АСУ. Описывается тестирование программного продукта с привидением листинга кода тестовых классов.

В заключении приводятся основные выводы, достигнутые в ходе выполнения работы.

В результате была разработана, описана и протестирована автоматизированная система управления рабочим пространством для тольяттинского офиса компании «NetCracker».

Выпускная квалификационная работа содержит пояснительную записку объемом 52 страницы, включая 20 иллюстраций, 5 таблиц, список литературы из 21 наименования, приложение.

ANNOTATION

The title of the graduation work is “Development of the Industrial Control System for Management of the Work Space in Togliatti Office of “Netcracker”.

This project deals with development of the industrial control system (ICS) for monitoring of workspace in an office.

The aim of the work is to develop a simple software with human-machine interface, which simplifies the management of the workspace.

The object of the project is the management of workspace.

The subject of the graduation work is ICS.

Administration Department and Human Resources Department of the company deal with management of workspace in an office basing on relevant regulatory documents. The departments also should have outline of the office. Questions and issues that connected with allocation of employees in an office are very important.

At the beginning of the graduation work were characterized the main processes of management of workspace.

In the project highlighted the issues of the estimation and filling of workstations and correct description of business processes, which occur in the company.

In the second part of this work were compared software development tools also presented flowcharts that describe realization of some methods.

In the last part were given conclusions that achieved during fulfilment of project.

As the result, was developed, described and tested the industrial control system for management of workspace in Togliatti office of “Netcracker”.

The graduation work consists of an explanatory note on 52 pages, including 20 figures, 5 tables, the list of 21 references, attachment.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Рабочее место и рабочее пространство — это специально организованные условия, в которых протекает трудовая деятельность человека. Рабочее пространство представляет собой зону, отделённую от природной среды с искусственно создаваемыми условиями, и является проектируемой частью рабочей среды. Офис, производственный цех, судоверфь – всё это примеры рабочего пространства. Рабочее место — это часть рабочего пространства, в котором располагается оборудование, с которым непосредственно взаимодействует человек в рабочей среде.

В 2008 году корпорация NEC провела объединение с компанией «Неткрэкер», которая работает над созданием, внедрением и сопровождением систем эксплуатационной поддержки (OSS), систем поддержки бизнеса (BSS), а также SDN/NFV-решений для крупных предприятий, операторов связи, и государственных учреждений.

Компания также предлагает услуги в области профессионального обслуживания (включая консалтинг, внедрение и поддержку) и сервисы по управлению телекоммуникационными процессами. Тольяттинский офис компании «Неткрэкер» на 2016 год насчитывает более 450 человек, почти все сотрудники находятся в деловом центре «Квадрат».

В Тольяттинском офисе «Неткрэкер» происходит непрерывная миграция сотрудников вместе со своими персональными компьютерами, и поэтому много времени уходит на поиск свободных мест, в том числе и для сотрудников, командированных из других офисов компании.

Учетом рабочих мест в компаниях в основном занимается отдел администрации и кадров. При учете рабочего пространства и рабочих мест используются соответствующие нормативные документы. Также на руках должен быть план-схема помещений.

Вопросы, касающиеся оптимизации и упрощения учета рабочего пространства и рабочих мест, являются очень актуальными. Актуальность темы

выпускной квалификационной работы связана со значительным распространением исследуемого явления и заключается в необходимости разработки рекомендаций по совершенствованию работы в рассматриваемой области.

Объектом исследования является процесс учета рабочего пространства и рабочих мест.

Предмет исследования: автоматизация процесса управления рабочим пространством.

Целью работы является разработка автоматизированной системы управления рабочим пространством для оптимального его распределения.

Задачи:

- проанализировать учебно-методическую и научную литературу необходимую для разработки автоматизированной системы управления;
- исследовать существующий процесс учета рабочего пространства и рабочих мест;
- разработать алгоритм учета рабочего пространства и рабочих мест для автоматизированной системы управления;
- реализовать алгоритм учета рабочего пространства и рабочих мест с помощью автоматизированной системы управления;
- определить эффективность разработанной автоматизированной системы управления в работе компании.

В первой главе проводится анализ деятельности отдела кадров и администрации офиса. Процесс учета рабочего пространства разбивается на этапы и по ним строится контекстная диаграмма всего процесса. Выделяются самые слабые этапы, нуждающиеся в автоматизации. Описываются основные принципы динамического программирования. На его основе ставится задача оптимального распределения сотрудников по офису и описывается её математическая модель.

Во второй главе описывается обоснование архитектуры проектируемой автоматизированной системы управления рабочим пространством. Описываются требования к системе по классификации требований FURPS+. Проводится обоснование выбора технологий для разработки и проектируется система. Представляются блок-схемы с изображением работы некоторых методов АСУ. Описывается тестирование программного продукта с привидением листинга кода тестовых классов.

В заключении приводятся основные выводы, достигнутые в ходе выполнения работы.

В результате была разработана, описана и протестирована автоматизированная система управления рабочим пространством для тольяттинского офиса компании «NetCracker».

Глава 1 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ ДЕЯТЕЛЬНОСТИ КОМПАНИИ ООО «НЕТКРЭКЕР»

1.1 Общая характеристика организации ООО «Неткрэкер»

Заказчиком данной работы является ООО «Неткрэкер». Одно из направлений, которым занимается компания – это создание, внедрение и сопровождение SDN и NFV решений для операторов связи.

«Неткрэкер» является дочерней компанией корпорации NEC. Центры разработки, технической поддержки и отделы продаж расположены по всему миру: Россия, Украина, Белоруссия, США, Индия. Также компания имеет свои офисы в Северной Америке, Европе, Южной Африке, Латинской Америке, на Ближнем Востоке и в Азиатско-Тихоокеанском регионе.

В 14 городах России, Белоруссии и Украины, «Неткрэкер» курирует учебные центры при вузах, потому что компания заинтересована в поиске молодых и талантливых студентов.

В 1993 году Майкл Файнберг и Бонни Ворд основали компанию «Неткрэкер». После создания компания благополучно развивалась и в 2008 году было принято решение об объединении с NEC Corporation. Компания «Неткрэкер» значительно расширила своё портфолио различными сервисными платформами, инновациями и продуктами, и профессиональными сервисами, благодаря этому объединению [13].

Компания получила масштабируемую и надежную технологию и добавила к своим комплексным решениям на сервисном и клиентском уровнях богатые функциональные возможности по активации сетевых ресурсов, благодаря бизнесу Subex, специализирующемуся на разработке решений по активации услуг, который компания «Неткрэкер» успешно приобрела в 2011 году [14].

Совместно с корпорацией NEC, в феврале 2015 года, компания «Неткрэкер», предприняла шаги к расширению совместного портфолио решениями в области виртуализации. В результате совместной инициативы

была создана новая структура и международный бренд, объединяющие многолетний опыт «Неткрэкер» в сфере телекоммуникаций и IT и инновационный подход к управлению сетями корпорации NEC. Объединенный бренд разрабатывает комплексные SDN/NFV - решения для различных сфер бизнеса [14].

Портфолио «Неткрэкер» включает продукты для поддержки всех сфер деятельности операторов и построено на основе единой платформы, поддерживающей облачные технологии: управления доходами, партнерами и клиентами, операциями, а также решения для управления инфраструктурой и виртуальными сетями. Портфолио включает продукты для виртуализации (SDN/NFV), систем поддержки бизнеса (BSS), систем эксплуатационной поддержки (OSS), систем управления взаимоотношениями с клиентами (CRM), биллинга, предоставления услуг (Service Fulfillment), обработки первичных учётных записей (Billing Mediation), тарификации (Rating), управления заказами (Order Management), домашней сети, управления линейно-кабельными сооружениями (Outside Plant), управления мобильными устройствами (Mobile Device Management), сетевого планирования, управления ИТ-услугами, управления конфигурациями (Configuration Management), планирования ресурсов (Network Resource Planning), и управления компьютерными сетями, облачных сервисов и аналитики больших данных (Big Data Analytics).

Тольяттинский офис компании «Неткрэкер» был основан 9 января 2008 года. Со временем офис увеличивался, появлялись новые отделы. Штат сотрудников тоже увеличивался. Необходимостью для администрации стал постоянный поиск нового рабочего пространства и рабочих мест. Изначально вся компания помещалась в одном кабинете, сейчас же сотрудники находятся на нескольких этажах и занимают около 40 кабинетов.

На рисунке 1 представлена схема организации тольяттинского офиса, деятельность которого будет рассмотрена.

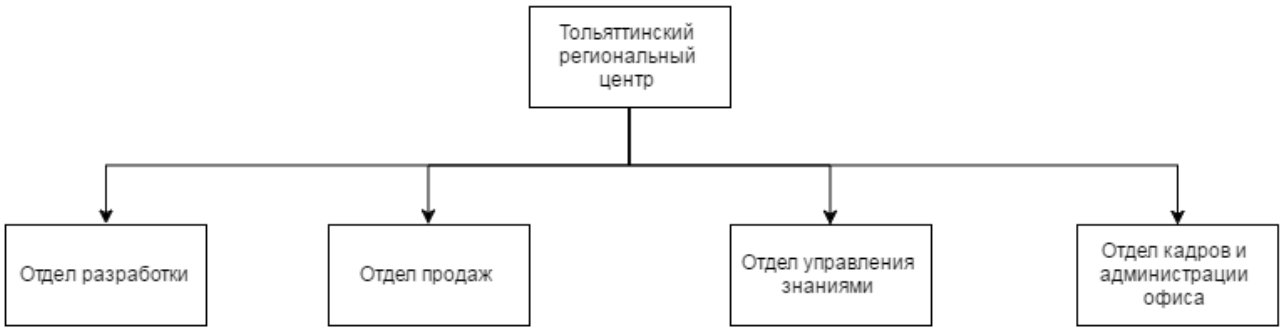


Рисунок 1 - Организационная структура предприятия

В дальнейшем будет рассматриваться деятельность только отдела кадров и администрации офиса. Учетом рабочего пространства занимается администрация офиса. Данному отделу требуется хранить всю имеющуюся информацию о рабочих местах, постоянно её изменять, проводить поиск и выдавать отчеты, чтобы повысить производительность труда и комфортность рабочей среды. Данные процессы являются трудоёмкими и нуждаются в автоматизации.

1.2 Анализ процесса учета рабочего пространства

На данный момент в тольяттинском офисе число сотрудников меняется изо дня в день, одни сотрудники уходят в командировку другие возвращаются, поступают новые сотрудники, некоторые команды растут и им требуется переезд в более просторный кабинет. Другими словами, происходит активная миграция сотрудников. В тоже время, администрации постоянно нужно знать, в каком состоянии находится рабочее пространство, есть ли в распоряжении свободные места, кабинеты и нет ли случаев, нерационального управления рабочим пространством.

Проанализируем процесс учета рабочего пространства, применяя структурный подход к моделированию. Для построения диаграмм воспользуемся программой Microsoft Visio Professional 2013 – векторным графическим редактором диаграмм и блок-схем для Windows. Данная

программа была выбрана, потому что Тольяттинский Государственный Университет предоставляет студентам бесплатную академическую подписку на полнофункциональную версию программного обеспечения, которая обладает всем необходимым функционалом для того, чтобы построить схемы процессов деятельности.

Для выполнения структурного анализа предприятия была использована методология функционального моделирования IDEF0 (Integrated Definition Function Modeling), согласно которой система представляется как «совокупность взаимодействующих процессов/работ/функций».

Ниже (рисунок 2), представлена контекстная диаграмма процесса учета рабочего пространства, на ней отображена входная и выходная информация, а также управляющие документы и механизмы, с помощью которых данный процесс может быть реализован.

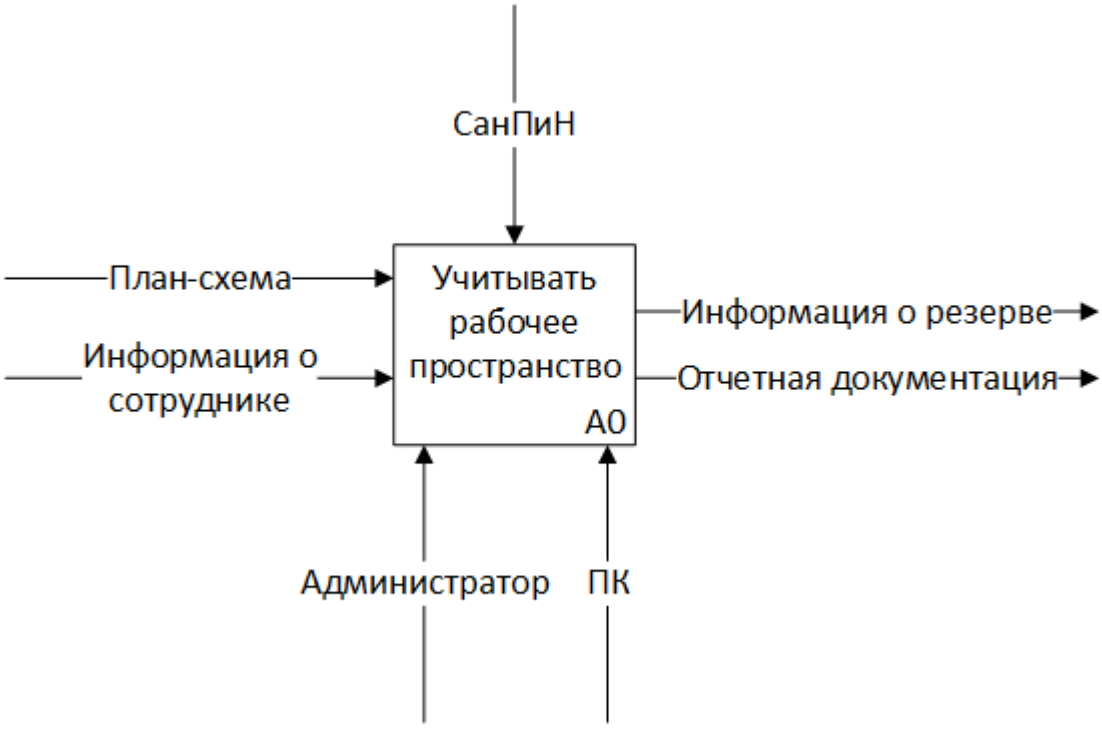


Рисунок 2 – Контекстная диаграмма процесса учета рабочего пространства

На рисунке 3 изображена декомпозиция процесса учета рабочего пространства.

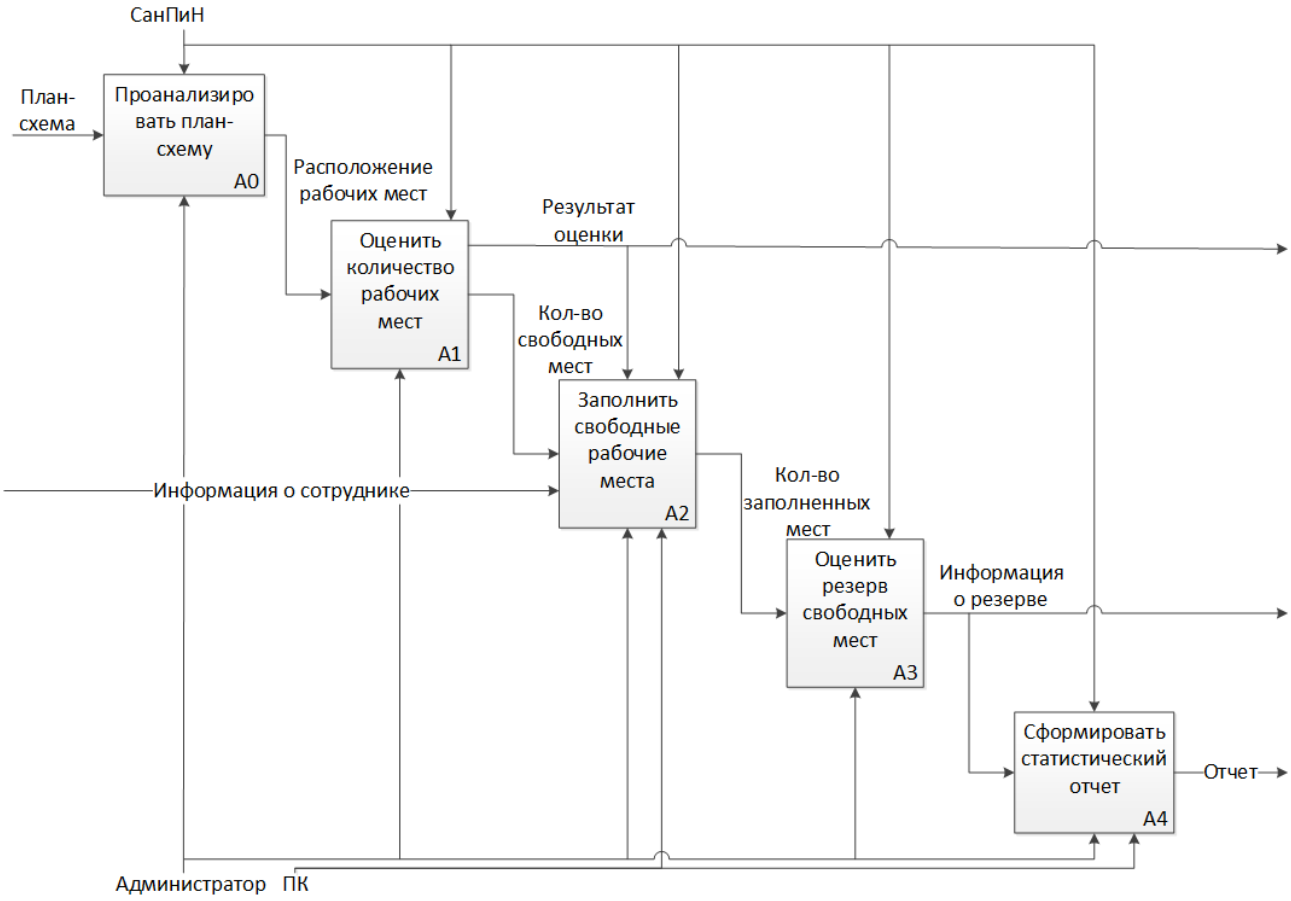


Рисунок 3 – Декомпозиция учета рабочего пространства

Узнать об изменении рабочего пространства отдел администрации может, проводя плановую проверку или непосредственно от сотрудников. Далее, руководствуясь нормативными документами, вносятся правки в текущую документацию и формируется отчет.

После описания системы в целом, проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиций. На рисунке 4 представлена контекстная диаграмма учета рабочего пространства при использовании АСУ. Красным цветом были выделены процессы, которые нуждались в оптимизации.

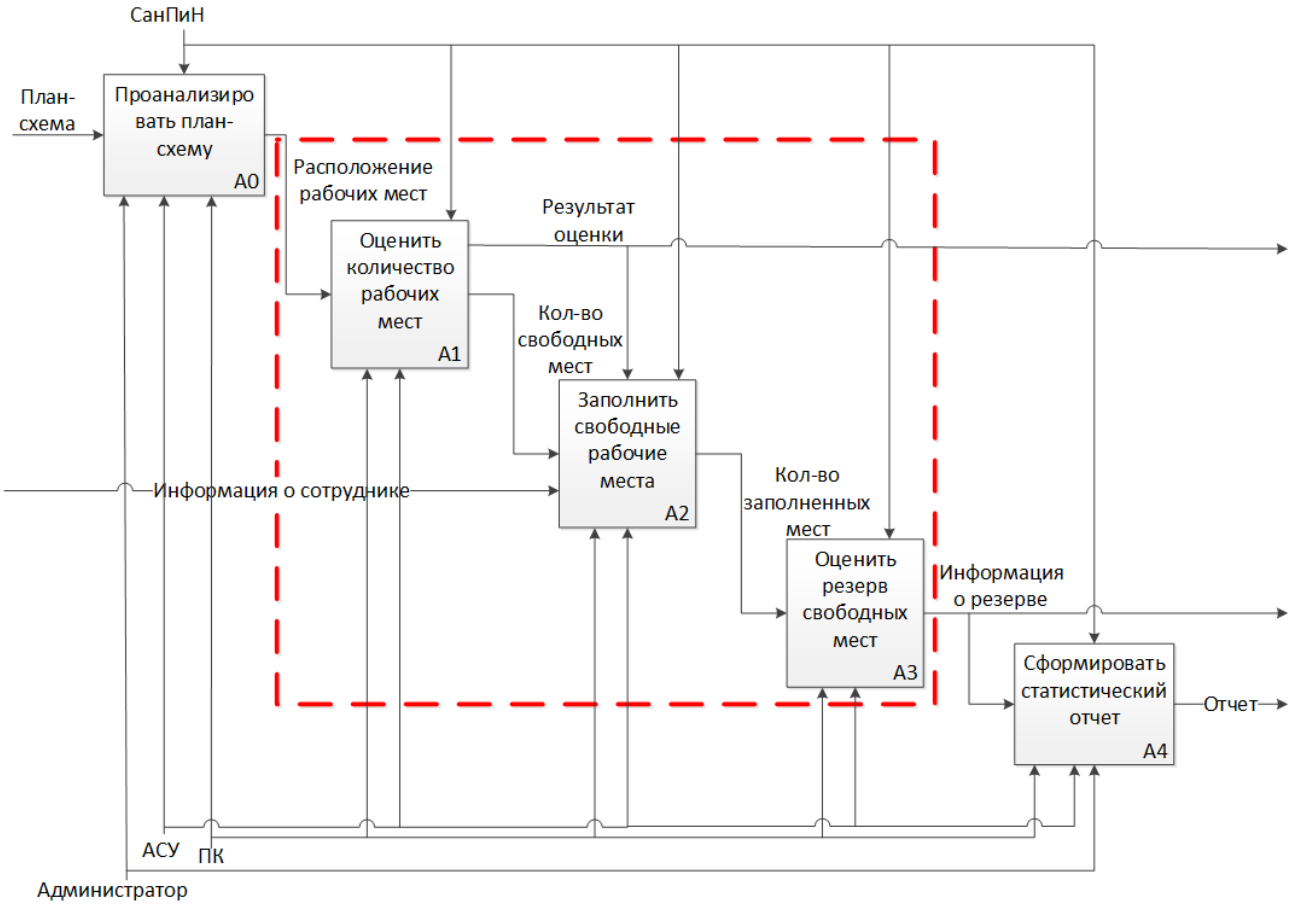


Рисунок 4 - Декомпозиция учета рабочего пространства с АСУ

Был рассмотрен процесс учета рабочего пространства, в нём можно выделить основные слабые места:

- отсутствие автоматизации бизнес процессов;
- плохая доступность информации для остальных сотрудников;
- высокие трудовые и временные затраты.

В результате анализа построенной диаграммы можно судить о том, что метод, используемый сотрудниками предприятия, устарел, а также требует больших трудовых и временных затрат. А это говорит о том, что необходима автоматизация процесса учета рабочего пространства, для которого необходимо разработать автоматизированную систему управления.

Внедрение автоматизированной системы управления позволит:

- оценить и улучшить качество заполнения площади офиса;

- позволит повысить производительность труда и комфортность среды функционирования офиса.

Таким образом, был рассмотрен существующий процесс рабочего пространства, далее необходимо формализовать требования к новому процессу и описать его.

1.3 Постановка задачи оптимизации рабочего пространства с помощью динамического программирования

В период с 1951 по 1953 годов благодаря работам Р. Беллмана возникло динамическое программирование. Принцип оптимальности Р. Беллмана лежит в основе динамического программирования, заключающийся в замене решения исходной многомерной задачи последовательностью задач меньшей размерности.

Принцип Р. Беллмана применяется для определения решений в многоэтапных процессах и выражается следующим типом: если в каждом из состояний будущее поведение системы не зависит от того, как она оказалась в этом состоянии, то дальнейшая траектория должна быть оптимальной. Под траекторией понимается последовательность состояний, в которых находится система [9].

В каждом состоянии системы известны воздействия, затраты и последствия на переход из одного состояния в другое. Пусть x_i шаговое управление на i этапе, $i = 1, n$, n – количество этапов. Решение задачи сводится к определению последовательности воздействий на состояние системы $X^*(x_1, x_2, x_3, \dots, x_n)$ при которой суммарные затраты минимальны:

$$\min \leftarrow Z = \sum_{i=1}^{n} Z_i, \tag{1.1}$$

где $Z_i -$ затраты на i шаге.

В соответствии с принципом оптимальности Р. Беллмана x_i выбираются таким образом, чтобы суммарные затраты на последующие этапы были

наименьшими (минимальными). Суммарные затраты зависят от состояния S и складываются из затрат на i шаге $z_i\ S, x_i$ и на последующих шагах. Суммарные затраты на все этапы обозначим Z_i, тогда оптимальное управление на каждом шаге определяется по рекуррентному уравнению динамического программирования [9].

$$Z_i\ S\ = \min\ z_i\ S, x_i\ + Z_{i+1}\ f\ S, x\ \ , \tag{1.2}$$

где $S -$ состояние системы;

$z_i\ S, x_i\ -$ затраты на i этапе;

$Z_{i+1}\ f\ S, x\ -$ затраты на следующем этапе.

Уравнение динамического программирования, являясь рекуррентным, показывает затраты на все оставшиеся этапы из любого состояния S через затраты на данном z_i и на всех последующих шагах $Z_{i+1}\ f\ S, x$. Для конечного шага затраты считаются по формуле:

$$Z_n\ S\ = \min_{x_n} z_n\ \ S, x_n\ , \tag{1.3}$$

где $z_n\ S, x_n\ -$ затраты на последнем шаге n.

Перед тем как перейти к описанию нового процесса с помощью АСУ, нужно разработать её математическую модель. Разработаем математическую модель динамического программирования оптимального размещения сотрудников по офису для системы управления рабочим пространством.

Постановка задачи: имеется конечное множество сотрудников и известна первоочерёдность и оценка эффективности (полезность) размещения сотрудников по офису, также имеется конечное множество кабинетов с известной вместимостью. Необходимо разместить сотрудников по офису таким образом, чтобы суммарный эффект (полезность) был максимальным.

Размещение сотрудников по офису – многоэтапный процесс.

Решение задачи определяется методом динамического программирования (метод Р. Беллмана).

Пусть $B_i -$ сотрудник, $i = 1, n$, $M_j -$ уникальное место размещения сотрудника ($M_0 -$ пустое рабочее место), $j = 0, m$, $X_{ij} -$ бинарная переменная ($X_{ij} = 1 -$ i сотрудник размещен на j месте, иначе $X_{ij} = 0$). Оценка эффективности (полезность) c_{ij} размещения B_i сотрудника на M_j месте размещения.

Математическая модель оптимального размещения сотрудников по офису имеет вид:

$$Z = \sum_{i=1}^{n} \sum_{j=0}^{m} c_{ij} * X_{ij}, \quad (1.4)$$

$$Z \to \max \quad (1.5)$$

Целевая функция при ограничениях:

$$\sum_{j=0}^{m} X_{ij} = 1 \quad (1.6)$$

$$\sum_{i=1}^{n} X_{ij} = 1 \quad (1.7)$$

$$X_{ij} \geq 0, \quad i = 1, n, \quad j = 0, m \quad 1.8$$

$$X_{ij} - \text{целое число} \quad (1.9)$$

Ограничение (1.6) обеспечивает размещение одного сотрудника на рабочем месте на i шаге.

Ограничение (1.7) обеспечивает уникальность места расположения сотрудника в офисе.

Ограничение (1.8) налагает неотрицательность на искомые переменные.

Ограничение (1.9) налагает дискретность на искомые переменные.

Опишем как работает алгоритм математической модели. Считается что, множество сотрудников $B = \{B_1, B_2, B_3, \dots, B_n\}$ изначально известно. Множество свободных рабочих мест $M = \{M_1, M_2, M_3, \dots, M_m\}$ определяется в результате предварительного анализа рабочего пространства. Далее происходит размещение множества сотрудников по множеству мест и оценивается эффективность размещения.

Оценка эффективности (полезность) c_{ij} размещения B_i сотрудника на M_j месте размещения, $i = 1, n,\ j = 0, m$ представляет показатель который может принимать значения из таблицы 1.1. Использующиеся обозначения в таблице:

- Ø – абсолютное отсутствие сотрудников в кабинете;
- - – сотрудники присутствуют, но не совпадают по отделу или проекту;
- + – сотрудники присутствуют и совпадают по отделу или проекту.

Таблица 1.1 - Значения критерия оптимальности

Совпадение		Критерий оптимальности c_{ij}
Проект	Отдел	
Ø	Ø	0,2
-	-	0,4
-	+	0,6
+	-	0,8
+	+	1

После определения множества оценок эффективности размещения, для каждого выбранного сотрудника будет выбрано размещение с максимальной полезностью, а остальные исключаются.

Важно отметить, что в задачах динамического программирования решение не всегда может быть единственным. В нашем случае может возникнуть ситуация, когда возможные размещения сотрудника по множеству свободных рабочих мест могут иметь одинаковую оценку эффективности, тогда не важно какое из них выбирать, полезность в данном случае будет одинаковой.

В результате проведенных исследований, была разработана математическая модель размещения сотрудников по тольяттинскому офису компании «Неткрэкер» на основе динамического программирования.

На рисунке 5 изображен алгоритм оптимального распределения сотрудников на основе построенной математической модели.

Рисунок 5 – Блок-схема алгоритма оптимального распределения сотрудников

Опишем последовательность выполнения алгоритма оптимального распределения сотрудников:

Шаг 1. Определить множество сотрудников B, участвующих в распределении.

Шаг 2. Провести предварительный анализ свободных рабочих мест.

Шаг 3. Определить множество свободных рабочих мест M, участвующих в распределении.

Шаг 4. Заполнить множество C возможными c_{ij} распределениями.

Шаг 5. Взять первый элемент множества C.

Шаг 6. Выделить подмножество $C_1 = (c_{11}, c_{12}, \dots, c_{1m})$ в котором он находился, если в подмножестве нет элемента большего чем текущий, тогда оставляем элемент, иначе исключаем его из множества C.

Шаг 7. Взять второй элемент множества C.

Шаг 8. …

Шаг N. Получено множество оптимальных распределений.

Была поставлена задача оптимального размещения сотрудников и описана её математическая модель. Для математической модели был разработан алгоритм работы.

1.4 Описание требований к разрабатываемой автоматизированной системе управления

На данный момент о том в офисе ли сейчас сотрудник или нет, в командировке или на больничном, знает только администрация, хотя эта информация публичная. Часто бывают ситуации, что человека ищут на рабочем месте, но не могут найти. Бывает трудно быстро найти свободную переговорную комнату, например, для незапланированного совещания.

Для улучшения качества рабочего процесса будет разработана автоматизированная система управления рабочим пространством с общим доступом, но это не значит, что редактировать её сможет каждый. Система будет защищена от вандализма путем авторизации пользователей. При запуске система будет доступна только в режиме чтения. Для перехода в режим

редактирования необходима авторизация. Данные от учетной записи (логин и пароль) будут выданы лично сотруднику, ответственному за учет рабочего пространства. В режиме чтения будет доступна вся информация обо всех сотрудниках компании в конкретном офисе. Будет отображаться план этажа, с его офисами и рабочими местами, каждый сотрудник сможет узнать, кто находится у него за стенкой или в каком кабинете и на каком месте сидит его знакомая. Гораздо быстрее получится узнать, есть ли сейчас свободная аудитория с вместительностью больше, если вдруг на занятие пришло больше сотрудников, чем запланировано.

Сейчас же все это делается «вручную», приходится самостоятельно обходить и обзванивать все возможные кабинеты в поисках сотрудника или места. При пополнении штата новыми сотрудниками или переездом команды разработчиков из одного помещения в другое, администрации приходится брать распечатанный план офиса и самостоятельно посещать кабинеты, где произошли изменения, записывать новых сотрудников, вычеркивать старых.

В автоматизированной системе управления данные процессы станут менее трудоёмкими. Для администрации офиса будут созданы специальные учетные записи, с правами на редактирование. Они смогут добавлять сотрудников, удалять помещения, при помощи пары нажатий кнопок. Наверное, самое главное это то, что будет реализовано автоматическое формирование отчетов. Также стоит подчеркнуть, что в автоматизированной системе управления не будет реализовано:

- учет мебели;
- учет техники;
- учет коммуникаций;
- учет заявок на ремонт и обслуживание.

Предполагается, что разработанная АСУ будет в открытом доступе для любого сотрудника тольяттинского офиса компании, поэтому система должна осуществлять проверку соответствия введённого пользователем логина и

пароля к учетной записи, логину и паролю в базе данных, так же необходимо реализовать механизм создания учетных записей только администраторов офиса. В деловом центре «Квадрат» не все кабинеты являются классическими рабочими, поэтому в системе необходимо реализовать возможность добавления новых конфигураций помещений с разными характеристиками:

- классическая;
- переговорная;
- открытое пространство.

Так же каждый кабинет должен иметь свои параметры:

- этаж;
- номер;
- вместительность.

В офисе может произойти такая ситуация, что один кабинет может перейти из-под контроля одной компании к другой, поэтому в системе обязательно нужно добавить возможность удаления помещений. Для получения информативных отчетов и для упрощения поиска будет реализована возможность добавления сотрудников с разными характеристиками:

- ФИО;
- E-Mail;
- тип сотрудника;
- командированный;
- постоянный.

В отличие от площади офиса, количество сотрудников может увеличиться, поэтому помимо функции удаления сотрудника будет добавлена функция добавления сотрудника. Формирование отчетов должно осуществляться по двум типам: по наличию свободных мест в офисе с привязкой к календарю или по размещению конкретного сотрудника. Далее необходимо перейти к анализу и последующему выбору технологий и средств для разработки АСУ, нам понадобится определится с языком

программирования, потом выбрать для него подходящий web-сервер и базу данных.

Вывод по главе 1

В первой главе выпускной квалификационной работы была проанализирована структура предприятия. В результате были выделены функции отдела администрации и выявлены процессы, требующие автоматизации.

Были собраны предпосылки к использованию автоматизированной системы управления и подробно изучен процесс учета рабочего пространства в тольяттинском офисе компании «Неткрэкер».

Были найдены проблемы данного процесса и их решение в виде автоматизированной системы управления. Была построена математическая модель задачи оптимального распределения сотрудников, на основе динамического программирования. Был описан новый процесс учета рабочего пространства с использованием АСУ.

Глава 2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ РАБОЧИМ ПРОСТРАНСТВОМ

2.1 Проектирование и обоснование архитектуры автоматизированной системы управления рабочим пространством

Была поставлена задача автоматизации процесса учета рабочего пространства, которая может быть решена путем разработки информационной системы управления рабочим пространством.

Разработка любой программы начинается с представления её архитектуры. Для автоматизированной системы управления рабочим пространством был выбран шаблон (паттерн) проектирования «Модель – Представление – Контроллер» (Model – View – Controller, MVC), изображенный на рисунке 6.

Основным преимуществом этого паттерна является минимальная зависимость, а иногда и вовсе её отсутствие между компонентами. Из этого следует, что все объекты поддерживают независимую модификацию. Изменив «Модель», нам не потребуется изменять «Представление» или «Контроллер».

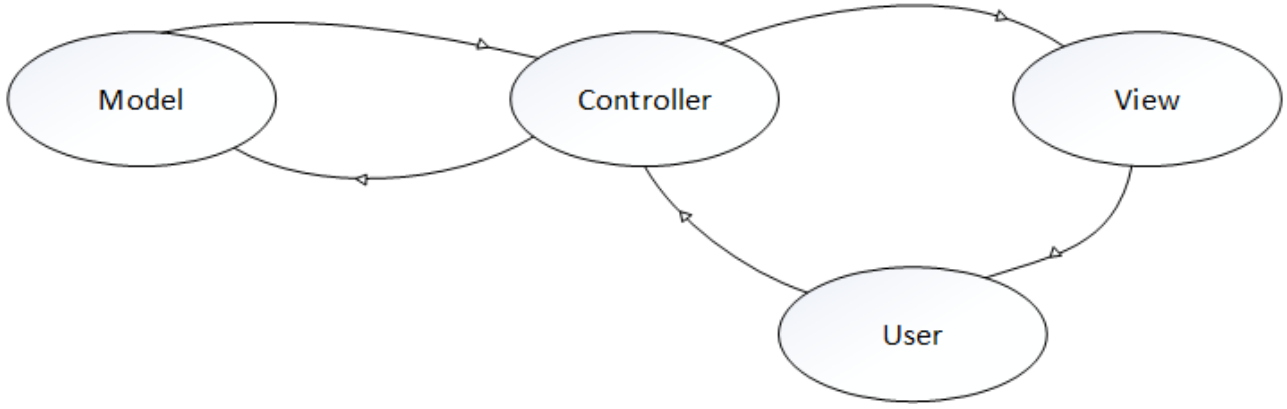


Рисунок 6 – Шаблон проектирования MVC

Разберём конкретно за что отвечает каждый компонент:

1. Model - включает в себя классы, которые отвечают за предоставление данных. Они реагируют на команды контролера и изменяют своё состояние;

2. Controller - состоит из классов, которые представляют действия пользователя, в нашем случае это будет администратор офиса. Пользователь выполняет действия в системе, которые обрабатываются контроллером и посылают модели команды для изменения;

3. View - определяет, как данные будут отображаться в системе, т.е. как их увидит конечный пользователь. Классы, которые входят в представление, должны реагировать на изменения моделей и изменять отображение информации.

4. User – показывает с какими модулями системы непосредственно работает пользователь (User видит View, использует Controller).

Для того, чтобы спроецировать программную архитектуру, созданную ранее, при проектировании, на исполняющую её физическую архитектуру системы. Необходимо описать физическое оборудование, на котором будет выполняться автоматизированная система управления рабочим пространством, а также описать, как она будет развертываться на это оборудование. Для этого была разработана диаграмма развертывания.

При разработке диаграмм развертывания преследуются следующие цели:

- спецификация физических узлов, необходимых для размещения на них исполнимых компонентов программной системы;
- отобразить физические связи между узлами реализации системы на этапе её исполнения.

Из диаграммы следует, что компонент Controller будет установлен на отдельном устройстве, которое изображено на рисунке как ServerPC. На стороне клиента (AdminPC) будут установлены компоненты Entity, зависимая от неё Math, а также View. Так же на ServerPC будет развернут сервер приложений и сервер базы данных.

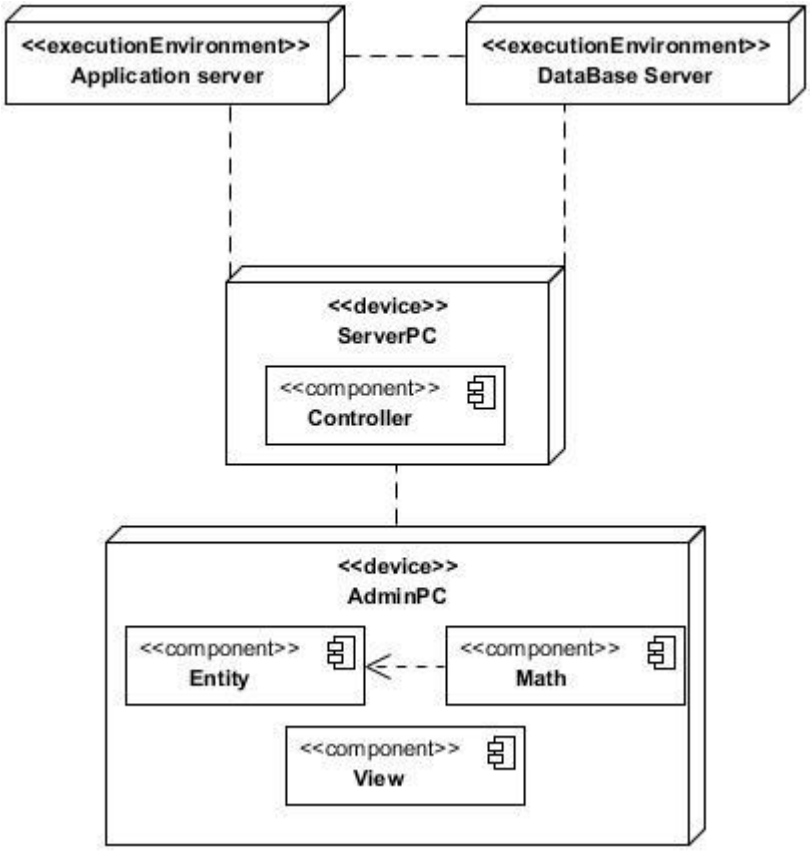


Рисунок 7 – Диаграмма развертывания системы

После описание выбранной архитектуры разработки определим основные требования, предъявляемые к будущей системе. Требования к системе описаны по методологии FURPS+ (таблица 2.1).

Таблица 0.1 - Требования к системе

ID	Требование	Статус	Полезность	Риск	Стабильность	Целевая версия
Functionality – Функциональные требования						
1	Отображать план-схему помещений	Одобренные	Важное	Средний	Низкая	1.0.0
2	Просматривать информацию о рабочем месте	Одобренные	Критическое	Средний	Средняя	1.0.0
3	Просматривать	Одобренные	Критическое	Средний	Средняя	1.0.0

ID	Требование	Статус	Полезность	Риск	Стабильность	Целевая версия
	информацию о сотруднике					
4	Предоставлять возможность редактировать данные	Одобренные	Важное	Средний	Низкая	1.0.0
5	Осуществлять составление отчетной документации	Одобренные	Критическое	Средний	Средняя	1.0.0
Usability – Требования к удобству использования						
6	План-схема должна быть контрастной	Одобренные	Важное	Низкий	Низкая	1.0.0
7	Размер текста 12 px	Предложенные	Важное	Низкий	Высокая	1.0.0
Reliability – Требования к надежности						
8	Доступ к системе 24/7/365	Одобренные	Критичное	Низкий	Низкая	1.0.0
9	Осуществление разграничения прав к системе	Предложенные	Критическое	Средний	Средняя	1.0.0
Performance – Требования к производительности						
10	Время отклика системы должно быть не более 250 мс	Одобренные	Важное	Высокий	Низкая	1.0.0
Supportability – Требования к поддержке						
11	Время устранения ошибок не более 2 часов	Предложенные	Важное	Средний	Средняя	1.0.0

ID	Требование	Статус	Полезность	Риск	Стабильность	Целевая версия
+ Ограничения						
Ограничения реализации						
12	АСУ должна быть совместима с Oracle Database	Одобренные	Критическое	Средний	Средняя	1.0.0
Ограничения форматов загружаемой информации						
13	Формат загружаемых план-схем офиса должен быть .svg	Предложенные	Важное	Средний	Средняя	1.0.0

После описания требований и выбора архитектуры будущего программного обеспечения можно приступать к выбору средств разработки.

2.2 Выбор средств разработки информационной системы управления рабочим пространством

2.2.1 Описание и сравнительный анализ языков программирования и выбор оптимального из них для разработки АСУ

Анализ начнется с имеющихся на данный момент языков и технологий создания веб-проектов, количество которых невелико, но каждый язык или технология в чем-то лучше другого и разработан для более узкой цели, чем другой. Приведем существующие и наиболее часто используемые языки программирования и технологии.

PHP — первоначально Personal Home Page Tools, сейчас Hypertext Processor, узко специализированный язык программирования для Web-разработки. Первый публичный релиз произошел 8 июня 1995 года, автором является Расмус Лердорф. Далее язык начал набирать популярность и последовали очередные релизы:

1. PHP 3 (1998 г.)
2. PHP 4 (2000 г.)
3. PHP 5 (2004 г.)
4. PHP 7 (2015 г.)

Сам язык испытал на себе влияние других, уже существовавших в то время языков программирования, таких как: Perl, C, C++, Java. В PHP цикл for each и ассоциативные массивы были заимствованы из языка Perl, а синтаксис подобен языку C.

Одним из главных достоинств языка является поддержка широкого набора баз данных, например, Adabas D, dBase, Empress, FilePro, Hyperwave, IBM DB2, Informix, Ingres, InterBase, FrontBase, mSQL, Direct MS-SQL, MySQL, ODBC, Oracle, Ovrimos, PostgreSQL, SQLite, Solid, Sybase, Velocis, Unixdbm. [1]

C# — язык объектно-ориентированного программирования, разработанный в период с 1998 по 2001 год. В его создании принимали участие группа инженеров под руководством Андерса Хейлсберга. Язык разрабатывался специально для разработки приложений для платформы Microsoft .NET Framework. В дальнейшем он был стандартизирован как ECMA-334 и ISO/IEC 23270. Язык C# обладает C-подобным синтаксисом, который является наиболее похожим на синтаксис таких языков, как Java и C++.

Язык поддерживает полиморфизм, имеет статическую типизацию, перегрузку операторов (в том числе операторов явного и неявного приведения типа), события, атрибуты, свойства, делегаты, методы и обобщённые типы, анонимные функции с поддержкой замыканий, итераторы, комментарии в формате XML, LINQ, исключения.

В C# исключены некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, т.к. языке опирается на практику своих предшественников, таких как:

1. C++.
2. Pascal.
3. Smalltalk.

4. Java.

Для примера можно привести множественное наследование классов, в сравнении с C++, C# не поддерживает множественное наследование классов, хотя допускается множественное наследование интерфейсов.

Java — является объектно-ориентированным языком со строгой типизацией данных. Язык бы разработан компанией Sun Microsystems, которая в дальнейшем была приобретена компанией Oracle.

Главное преимущество языка Java заключается в том, что язык при компиляции транслируется в специальный байт-код, поэтому приложения, написанные на этом языке, могут работать на любой компьютерной архитектуре, с помощью виртуальной Java-машины. Программный код на языке Java транслируется в байт-код, который уже выполняется виртуальной машиной Java (JVM, Java Virtual Machine). JVM — специальная программа, для обработки байтового кода и передачи инструкций оборудованию, похожая на интерпретатор.

Полная независимость байт-кода от операционной системы и оборудования, это главное достоинство подобного способа выполнения программы. Если для устройства существует соответствующая JVM, то данный способ позволит выполнить на ней Java-приложение.

Гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной, является ещё одной важной особенностью технологии Java. Немедленное прерывание программы может быть вызвано любой операцией, которая превышает установленные полномочия программы. Например, соединение с другим компьютером или попытка несанкционированного доступа к данным.

Наверное, единственным недостатком концепции использования JVM, является снижение производительности. По данным сайта shootout.alioth.debian.org, для семи разных задач время выполнения на Java составляет в среднем в полтора-два раза больше, чем для C/C++.

Проведем сравнение языков программирования в таблице 2.2

Таблица 2.2 – Сравнение языков программирования

Язык программирования	Высокая скорость выполнения (0-1)	Интерпретатор для всех ОС (0-1)	Знание языка разработчиком (0-5)	Итог
PHP	1	1	1	3
C#	1	0	3	4
Java	1	1	5	7

Проведя сравнительный анализ существующих языков для разработки используемых для создания веб-проектов, были выбраны три кандидата: PHP, Java и C#. Изучив существующую документацию по языкам и сравнив их слабые и сильные стороны сделан вывод, что наиболее подходящим языком для разработки автоматизированной системы управления рабочим пространством является Java.

2.2.2 Описание и сравнительный анализ веб-серверов и выбор оптимального из них для разработки АСУ

Так как система будет представлять собой веб-приложение, то при его разработке требуется использовать веб-сервер.

Apache Tomcat— контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов и спецификацию Java Server Pages (JSP) и Java Server Faces (JSF). Написан на языке Java. Tomcat позволяет запускать веб-приложения, содержит ряд программ для самоконфигурирования. Используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish.

WildFly — Java EE-сервер приложений с открытым исходным кодом. WildFly можно свободно загрузить и использовать. Расширенная реализация принципов Java EE делает WildFly конкурентом для аналогичных программных решений, таких, как WebSphere или WebLogic. В качестве контейнера сервлетов JBoss AS использовал Tomcat. WildFly использует Undertow.

Cherokee — свободный кроссплатформенный веб-сервер, написанный на Си и поддерживающий современные технологии, включая FastCGI, SCGI, PHP, CGI, SSI, HTTPS (TLS и SSL), виртуальные хосты, балансировку нагрузки и другие [1]. Расширяем, благодаря поддержке плагинов. В основе лежит реализация принципов высокой производительности и скорости работы. Наличие веб-интерфейса позволяет упростить настройку. В ряде тестов показывает более высокую скорость работы, в сравнении с такими серверами, как Lighttpd и Nginx[1].

Проведем сравнение веб-серверов в таблице 2.3:

Таблица 2.3 – Сравнение веб-серверов

Веб-сервер	Поддержка Java EE 7	Бесплатная лицензия	Опыт работы разработчика с веб-сервером	Итог
Apache Tomcat	-	+	+	2
WildFly	+	+	+	3
Cherokee	-	+	-	1

На данный момент существует большое множество веб-серверов, которые можно использовать в качестве разработки ПО. Изучив существующий выбор, были выбраны три веб-сервера Apache Tomcat, WildFly и Cherokee, т.к. они являются наиболее популярными среди разработчиков. Все они имеют бесплатную лицензию, что является положительным качеством при выборе веб-сервера. Однако, поддержку стандарта Java EE 7 имеет только веб-сервер

WildFly, он был выбран для разработки автоматизированной системы управления.

Выше были выбраны основные инструменты разработки автоматизированной системы управления рабочим пространством, далее можно приступать к моделированию будущей системы.

2.3 Моделирование автоматизированной системы управления

Процесс моделирования функций, выполняемых рассматриваемой информационной системой, путем создания описательного структурированного графического изображения, показывающего что, как и кем делается в рамках функционирования объекта, связывающих эти функции с учетом имеющейся информации, называется – функциональное моделирование [2].

Язык объектного моделирования UML и программа Visual Paradigm 14.1 отлично подходят для осуществления функционального моделирования.

Visual Paradigm - это система управления требованиями, поддерживающая полный цикл разработки программного продукта. Visual Paradigm обеспечивает поддержку версионности и одновременной работы команды пользователей над одним проектом. В основе моделирования в Visual Paradigm лежит UML – язык графического описания для объектного моделирования, моделирования бизнес-процессов, системного проектирования и отображения организационных структур [3].

Функциональное моделирование с использованием UML направлено на выделение функций, которые будет выполнять информационная система. Для этого необходимо построить диаграмму вариантов использования, на которой изображаются отношения между актерами и вариантами использования.

Диаграмма вариантов использования необходима для следующих целей:

- определение контекста моделируемой предметной области на начальных этапах проектирования системы;

- формулирование общих требований к функциональному поведению системы;
- разработка исходной модели системы для её последующей детализации в форме логических моделей.

Назначение данной диаграммы состоит в следующем: проектируемая программная система представляется в форме вариантов использования (прецедентов), с которыми взаимодействуют внешние сущности или актеры [4].

В таблице 2.4 приведено краткое описание прецедентов функциональной модели вариантов использования

Таблица 2.4 – Краткое описание прецедентов

Прецеденты	**Описание**
Просмотр план-схемы офиса	Ознакомление с информацией о пространстве
Поиск сотрудника	Поиск сотрудника определённому параметру на схеме
Изменение рабочего пространства	Добавление, удаление, изменение конфигурации помещений и т.п.
Формирование отчетов	Составление отчетной документации о текущем состоянии рабочего пространства и о планируемых изменениях.

На рисунке 8 изображена диаграмма вариантов использования АСУ. На данной диаграмме выделены два актера. Актер «Сотрудник» имеет два варианта использования: «Просмотр план-схемы» и «Поиск сотрудника». Актер «Администратор» имеет пять варианто использования: «Просмотр план-схемы», «Поиск сотрудника», «Добавить сотрудника», «Удалить сотрудника» и «Формирование отчетов».

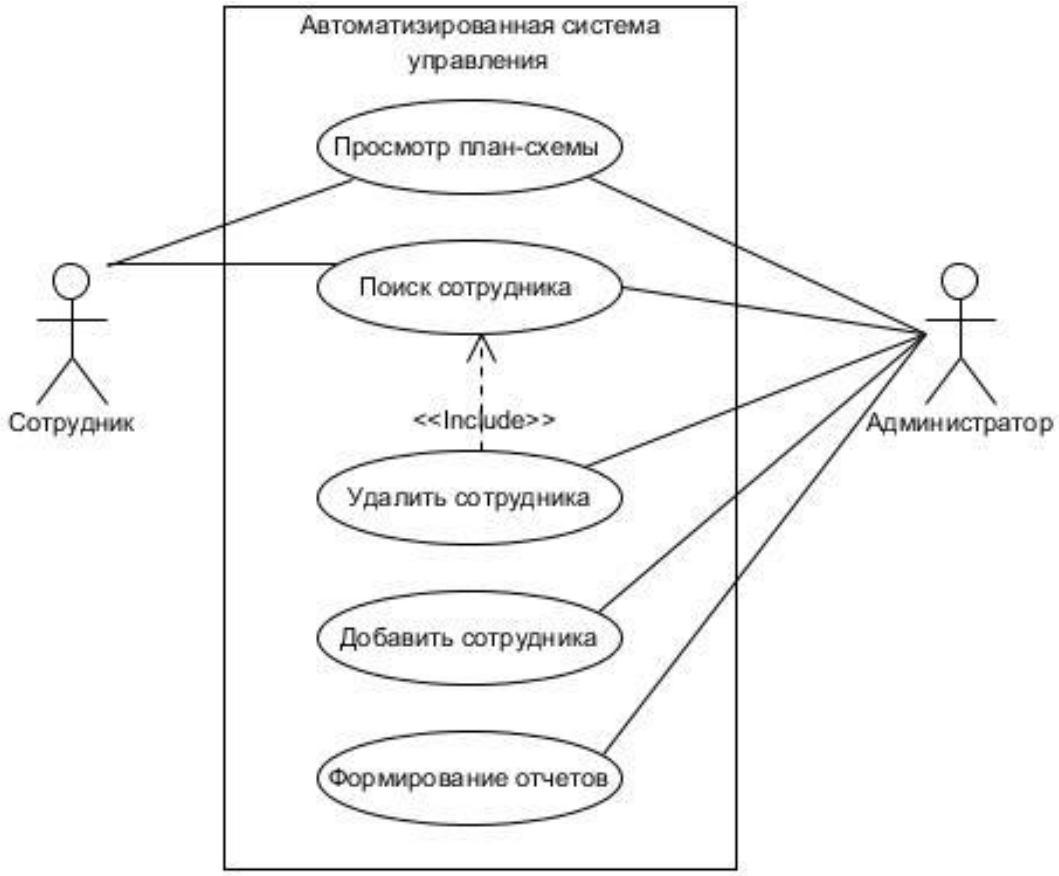


Рисунок 8 - Диаграмма вариантов использования АСУ

С помощью, разработанной модели вариантов использования, мы отобразили границы моделируемой предметной области и сформулировали общие требования к функциональному поведению проектируемой системы.

На рисунке 9 представлена диаграмма классов автоматизированной системы управления. В системе класс WebController отвечает за оповещение модели о необходимости изменений, классы Employee, Room и Math представляют собой данные и реагируют на сигналы контроллера изменяя свое состояние. Класс View отвечает за отображение данных пользователю и реагирует на изменение моделей. Информация из базы данных передаётся в классы Employee и Room, т.к. они являются классами сущностей из базы данных. Тип связи класса Math с классами Employee и Room – композиция. Классы WebController и View по отношению к классам Employee, Math и Room имеют тип связи ассоциация.

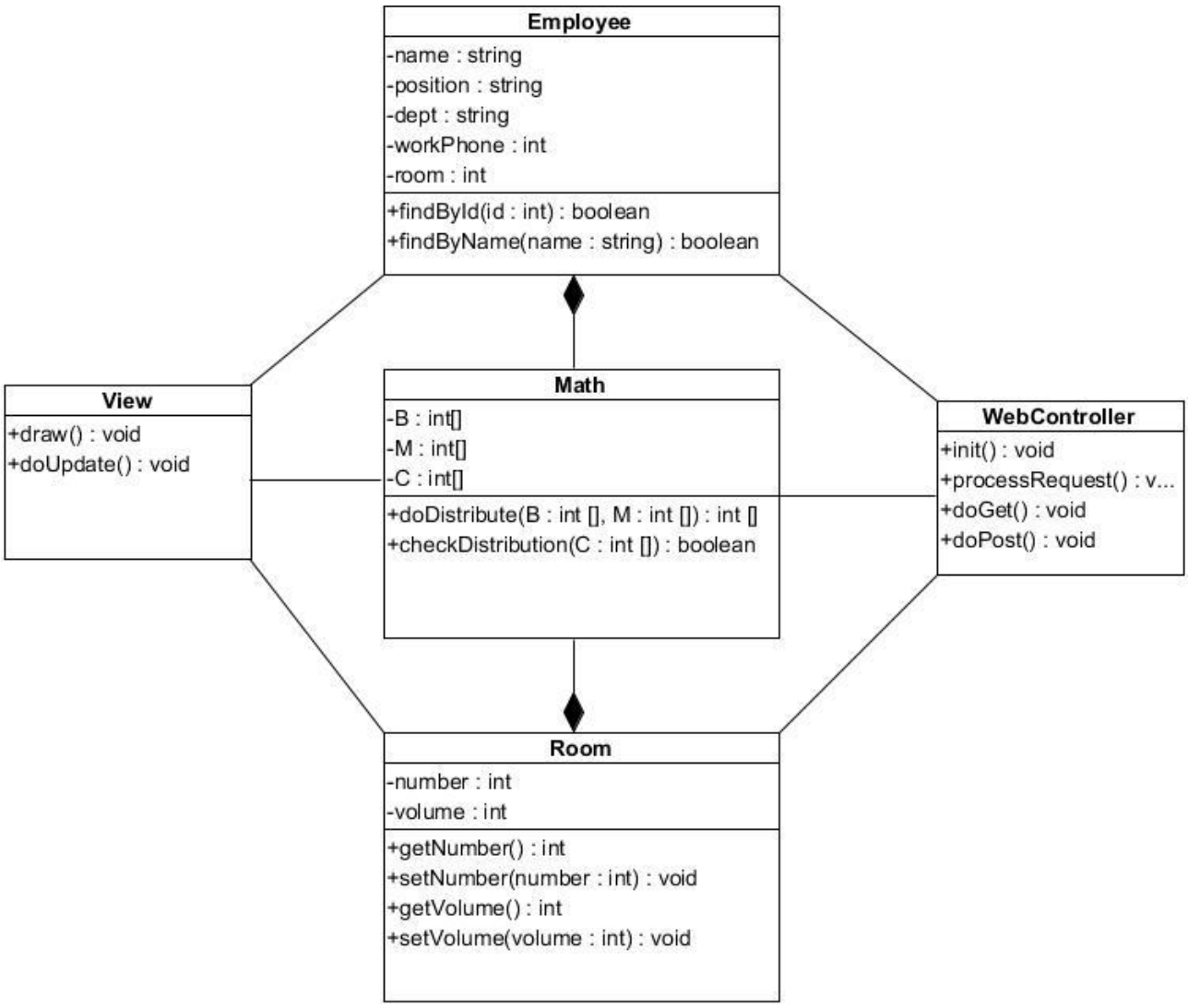


Рисунок 9 – Диаграмма классов системы

На рисунке 10 изображена логическая хранения данных. В модели имеется пять сущностей: «Сотрудник», «Рабочее место», «Кабинет», «Тип кабинета» и «Тип должности».

Сущность «Сотрудник» имеет первичный ключ «ID сотрудника». «Сотрудник» связан с сущностью «Рабочее место», мощность связи 1:1.

Сущность «Рабочее место» имеет первичный ключ «ID рабочего места». «Рабочее место» связано с сущностью «Кабинет», мощность связи 1:1.

Сущность «Кабинет» имеет первичный ключ «ID кабинета».

Сущность «Тип должности» имеет первичный ключ «ID должности». «Тип должности» связан с сущностью «Сотрудник», мощность связи 1:M.

Сущность «Тип кабинета» имеет первичный ключ «ID типа кабинета». «Тип кабинета» связан с сущностью «Кабинет», мощность связи 1:M.

Построение логической модели хранения данных было необходимо для проверки логики АСУ на наличие ошибок.

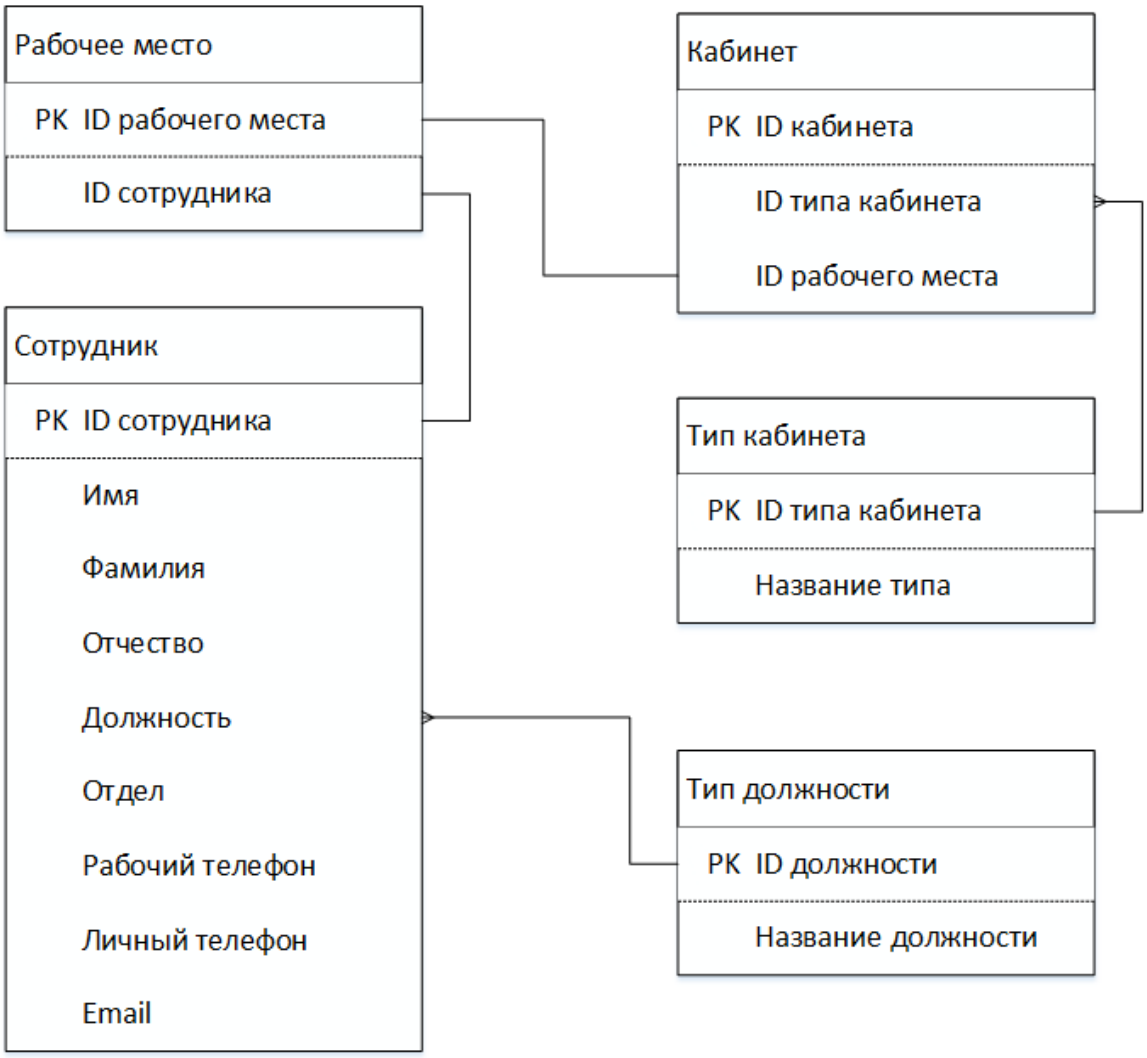


Рисунок 10 – Логическая модель хранения данных

Таким образом, после завершения построения логической модели хранения данных нужно построить физическую модель хранения данных для непосредственного отображения таблиц и полей базы данных.

Физическая модель хранения данных представлена на рисунке 11. Таблица Employee содержит информацию о сотрудниках и имеет ключевое поле ID employee. Таблица Room описывает какие конкретные рабочие места в ней находятся и также информирует о типе данного кабинета. Таблица

Workplace показывает какой сотрудник находится на конкретном рабочем месте, поле ID workplace является ключевым. Так же в базе данных существуют две вспомогательные таблицы, под названиями Room_type и Position_Type, которые помогают более подробно описать сотрудника и кабинет.

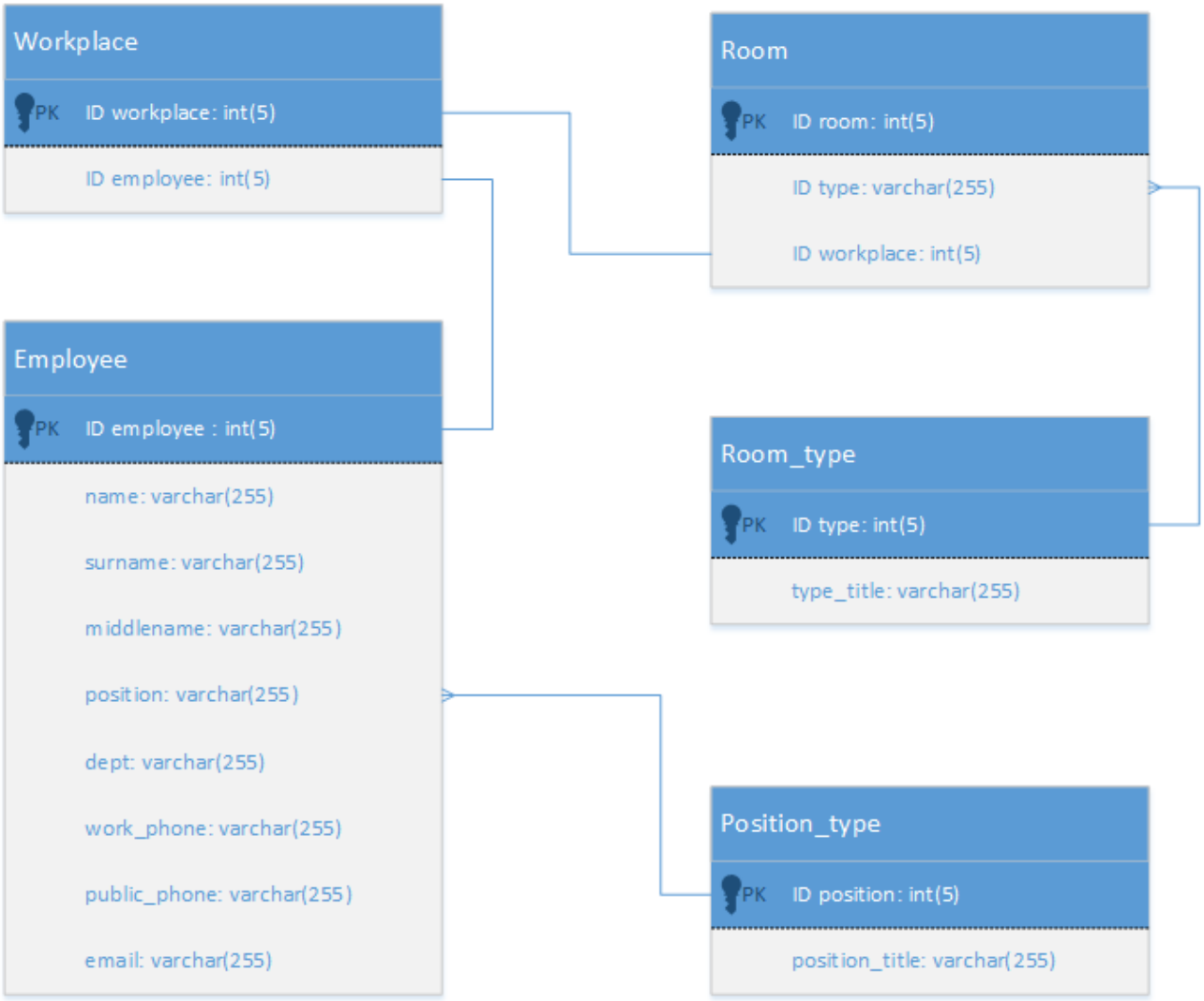


Рисунок 11 – Физическая модель хранения данных

Теперь имеется всё необходимое для реализации АСУ. Описаны основные прецеденты, построена диаграмма вариантов использования и диаграмма классов при помощи языка графического описания моделирования UML. Поскольку в системе существует хранилище данных, то была описана модель их хранения. Можно сказать, что моделирование автоматизированной системы управления рабочим пространством завершено и можно приступить к реализации программных модулей.

2.4 Реализация программных модулей автоматизированной системы управления рабочим пространством

Программным модулем называется функционально завершенный фрагмент программы. Ниже перечислены его основные характеристики:

- в модуль есть только один вход где он принимает набор исходных данных и один выход из него, где он возвращает один набор выходных данных;
- модуль содержит набор классов достаточный для необходимой обработки данных. Можно сказать, что модуль должен иметь функциональную завершенность;
- другие модули программы не должны влиять на результат работы данного модуля (фрагмента программы);
- модуль не должен быть перегружен, его размер должен быть в разумных пределах.

После введения понятия модуля программы, следует описать диаграмму пакетов системы (рисунок 12).

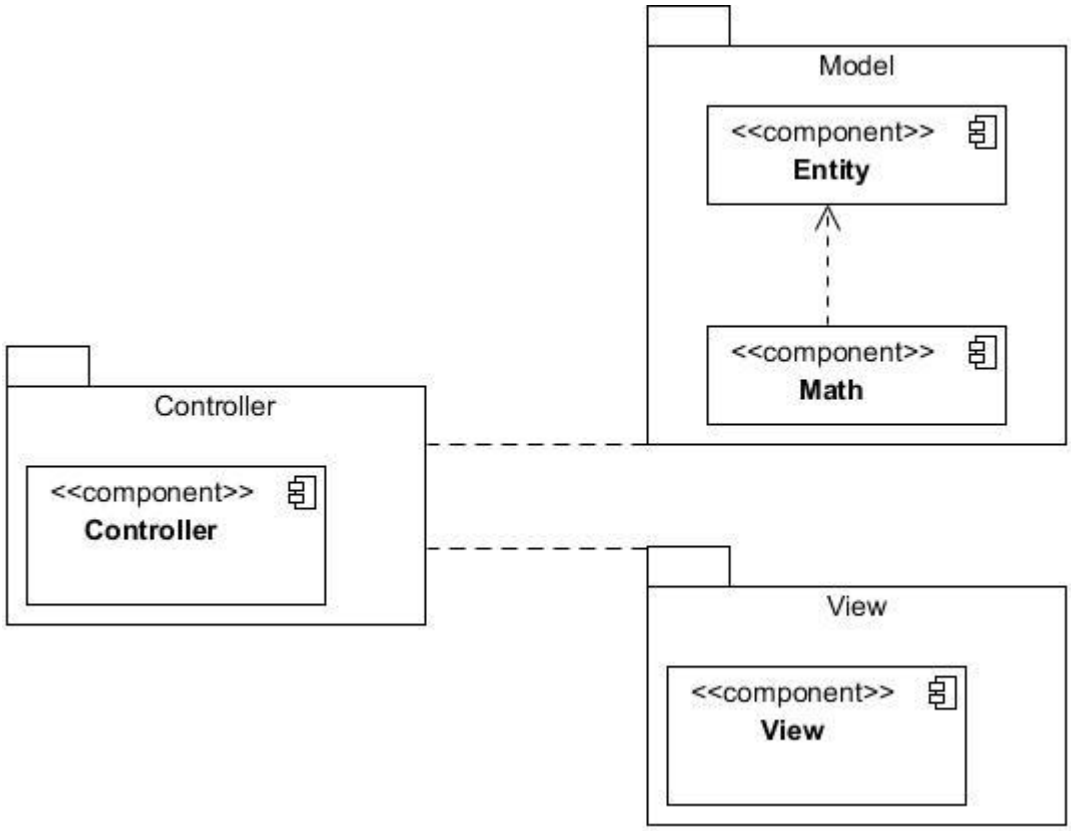


Рисунок 12 – Диаграмма пакетов

В соответствии с паттерном проектирования MVC, классы в программе были разбиты на три основных пакета:

1. View.
2. Controller.
3. Model.

Пакеты View и Controller отвечают за содержание в себе компонент представления и контроллера. В пакете Model находятся две компоненты Entity и Math. Компонента Entity отвечает за работу классов сущностей из базы данных (Employee и Room). Компонента Math отвечает за выполнение математической модели автоматизированной системы управления рабочим пространством.

При создании программного модуля следует соблюдать следующие рекомендации относительно его размера, сцепления с другими модулями и рутинности.

В первую очередь нужно следить за количеством и размером модулей программы. Разрабатывая много маленьких модулей, усложняется общая структура и схема программы. Разрабатывать большие модули нежелательно из-за того, что они:

- сложны для понимания;
- неудобны для изменений;
- замедляют работу и отладку программы.

Чем выше прочность внутренних связей программных модулей, тем проще сама программа. Существует несколько степеней прочности программных модулей:

- прочность по совпадению. Модуль составлен из повторяющихся в нескольких местах программы последовательности операторов. Этот вид прочности характерен для модульного программирования. Часто реализуется с помощью директив препроцессора и операторов транслятора, таких как #include в C# иimportв Java. Не рекомендуется к использованию;

- функциональная прочность. Модуль реализует какую-либо определенную функцию, однако при этом может использовать и другие модули. Данная степень прочности свойственна модульному и объектно-ориентированному программированию. Рекомендуется к использованию;
- информационная прочность. Модуль выполняет несколько операций над одной и той же структурой данных, область видимости которой ограничена самим модулем. Каждая операция имеет свою точку входа и свою форму обращения к данным. Является высшей степенью прочности и рекомендуется к использованию.

Мера зависимости модуля от других модулей по способу передачи данных называется - сцеплением модуля. При ослабевании сцепления модуля с другими модулями, усиливается его независимость от этих модулей.

На данный момент рекомендуется использовать параметрическое сцепление (сцепление по данным), хотя существует несколько других способов. При данном виде сцепления, данные передаются модулю двумя способами:

1) в виде значений его параметров при обращении к модулю;
2) как результат обращения к другому модулю для вычисления некоторой функции.

Этот вид сцепления реализуется в языках программирования при обращении к подпрограммам.

Независимость от предыстории обращений к модулю, называется – рутинностью модуля. Если результат обращения к модулю не зависит от результатов предыдущих обращений и зависит только от значений его параметров, тогда модуль считается рутинным. Не разрешается использовать глобальные переменные для последующих обращений, передачу значений через статические переменные или рекурсию. Если результат обращения к модулю зависит от его внутреннего состояния, то модуль считается зависящим от предыстории. К такому состоянию приводит, например, использование, статических переменных или рекурсии. Такие модули не рекомендуется

использовать, так как в некоторых случаях они могут спровоцировать ошибки, которые трудно будет обнаружить в силу привлечения скрытых механизмов, основанных, например, на использовании стека. Но, к сожалению, не всегда получается избежать использования таких модулей. Для достижения некоторых поставленных целей использование модулей, зависимых от предыстории является необходимым. В пример можно привести использование скриптов в web-страницах для разных звуковых и визуальных эффектов.

Отсюда напрашивается вывод, что на использование модулей, зависящих от предыстории нет строгого запрета, просто нужно исходить из следующих рекомендаций:

- всегда использовать рутинный модуль, если он имеет параметрическое сцепление;

использовать модуль, зависящий от предыстории, если это необходимо для обеспечения параметрического сцепления. При этом, в его спецификации должна быть четко сформулирована данная зависимость, так, чтобы была возможность прогноза поведения модуля при последующих обращениях. Например, использовать итерацию для поиска решений методом последовательных приближений или для реализации мерцающих изображений на Web-страницах.

Ниже на рисунке 13 представлено как отображается кабинет с рабочими местами в системе.

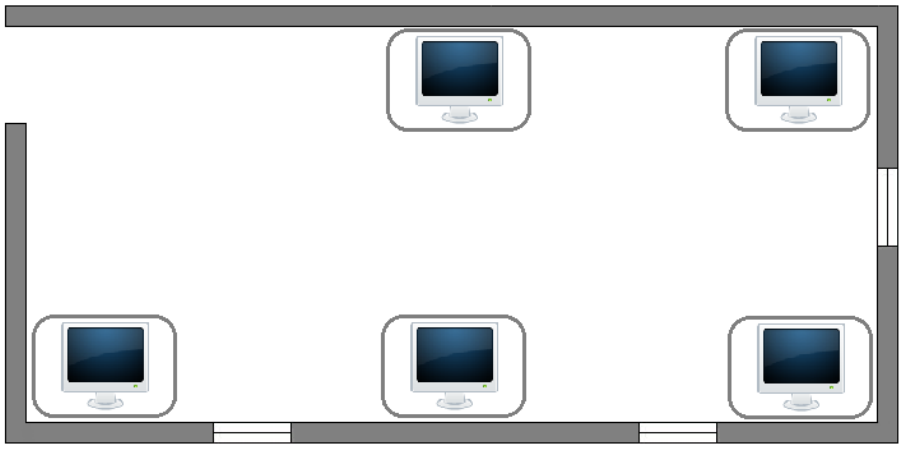


Рисунок 13 - Представление кабинета в системе

Данная схема является тестовой и создана только в качестве примера. На схеме каждое рабочее место отмечается серым прямоугольником с закруглёнными краями и вписанным в него специальной иконкой монитора.

При нажатии на объект представления рабочего места отобразится карточка сотрудника (рисунок 14), которому оно принадлежит. На карточке сотрудника присутствует информация двух видов личная (ФИО, должность, отдел) и контактная (рабочий телефон, мобильный телефон, почта, кабинет).

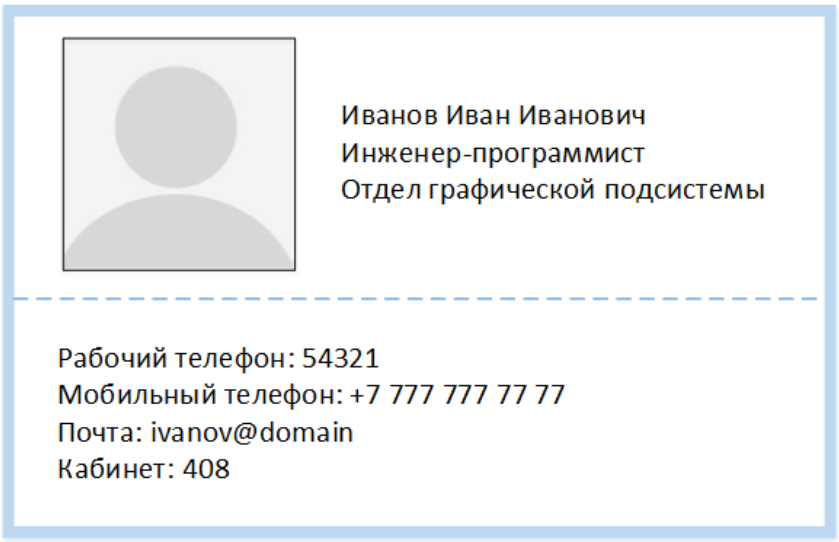


Рисунок 14 – Отображение карточки сотрудника в системе

На рисунке 15 представлена блок-схема метода на добавление сотрудника.

На первом этапе происходит стандартная процедура ввода и проверки корректности данных введённых пользователем, в нашем случае этим пользователем будет сотрудник отдела кадров и администрации офиса.

Далее если данные валидны, т.е. корректны система произведёт поиск возможного места и спросит у пользователя подтверждения, если всё верно, произойдет сохранение данных и метод завершится.

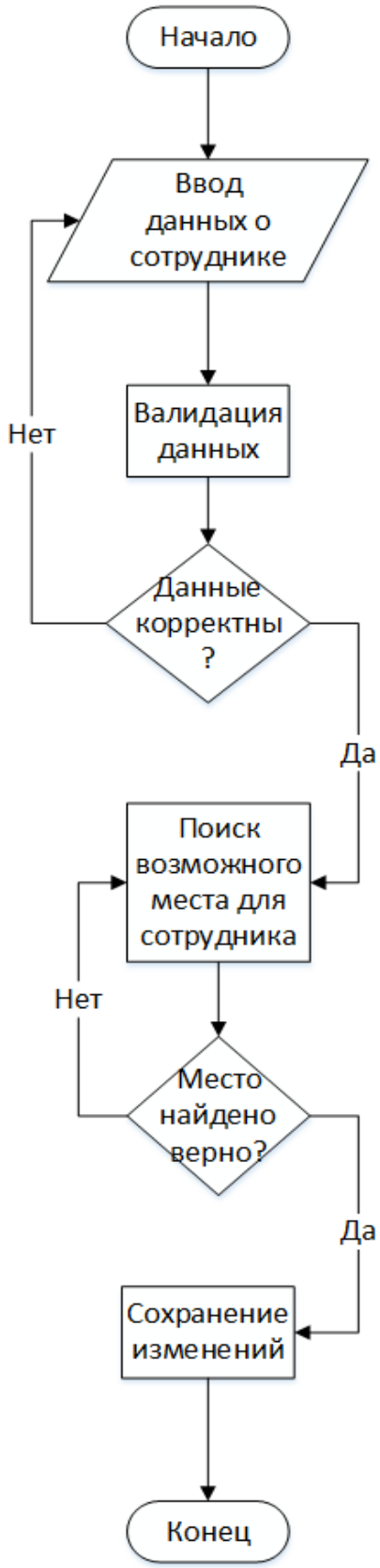
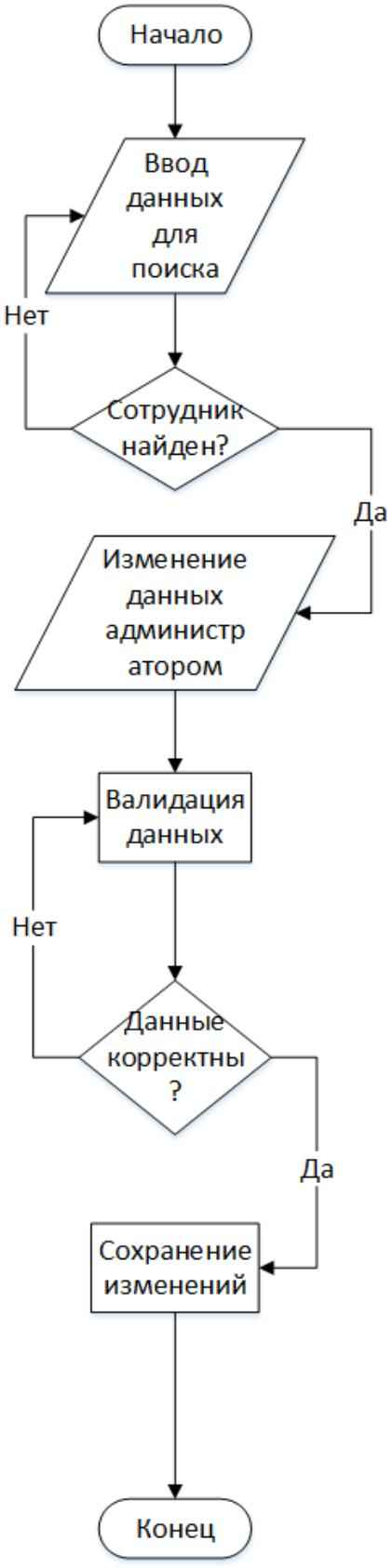


Рисунок 15 - Блок-схема реализации метода «Добавить сотрудника»

На рисунке 16 представлена блок-схема метода на изменение данных сотрудника.



Вначале вводятся необходимые данные для поиска, это может быть либо ФИО сотрудника, либо уникальный пятизначный телефонный номер, который выдаётся каждому сотруднику компании. Далее происходит поиск, если сотрудник найден, то администратору отобразиться вся информация о сотруднике и можно будет внести необходимые правки, иначе произойдет возврат к вводу данных. После ввода данных произойдет валидация, если всё в порядке, то данные сохранятся в системе и вступят в силу.

Рисунок 16 - Блок-схема реализации метода «Изменить данные»

Кроме изменения рабочего пространства система может составить отчет, в котором будет доступна информация о количестве сотрудников по отделам и в сумме на определенный момент времени. Отчет самостоятельно сформированный системой представлен ниже на рисунке 17. Все данные были выдуманы и являются тестовыми. На рисунке представлено количество сотрудников как общее, так и по отделам в частности, также присутствует информация о резерве рабочих мест.

	Период	01.03.2017	01.04.2017	01.05.2017
Количество сотрудников	Отдел разработки	13	15	15
	Отдел продаж	5	4	5
	Отдел управления знаниями	4	4	4
	Отдел кадров и администрации офиса	5	7	7
Всего		27	30	31
Резерв рабочих мест		10	7	6

Рисунок 17 – Пример отчета в системе

На рисунке 18 изображено представление возможных вариантов распределения, при добавлении нового сотрудника в систему.

Сотрудник	Возможное расположение	
	Кабинет	Место
Иванов И.И.	408	144
	312	109
	313	115

Рисунок 18 - Пример распределения сотрудника в системе

Листинг программного кода, формирующего содержание таблицы с оптимальными размещениями, приведен в приложении А.

В данном разделе была представлена реализация автоматизированной системы управления рабочим пространством, далее перейдем к тестированию системы.

2.5 Тестирование разработанной автоматизированной системы управления рабочим пространством

Поскольку архитектура разработанной АСУ базируется на использовании множества сильно связанных компонентов, которые слабо сцеплены друг с другом, логичным является применить подход к тестированию по степени изолированности, а именно модульное тестирование (unit testing). Подход модульного тестирования позволяет проверить корректность отдельных модулей исходного кода программы.

Основная задача модульного тестирования заключается в написании тестов для каждого нетривиального метода или функции. Данный подход позволяет быстро проверять поступившие изменения кода на правильность, привело изменение кода к новым ошибкам или нет.

Для применения модульного тестирования на Java была использована специальная библиотека JUnit.

Ниже приведен пример модульного тестирования рабочих функций системы. На рисунке 19 представлен листинг кода метода testFindEmployeeById.

Данный метод проверят работу метода findById класса Employee. Метод помечен специальной аннотацией @Test, которая означает что данный метод является тестовым. Далее создаётся экземпляр класса Employee и вызывается метод findById со значением 32499. Данное значение было выбрано специально т.к. соответсвует значению из базы данных и является уникальным номером конкретного сотрудника, соответственно мы ожидаем что переменная isFinded примет значение true. Если ожидаемое значение совпадёт с пришедшими, то

тест будет считаться пройденным и программа скомпилируется, иначе мы получим ошибку об аварийном завершении компиляции с указанием на провалившийся тестовый метод.

```
@Test
public void testFindEmployeeById() {
    Employee employee = new Employee();
    boolean isFinded = employee.findById(32499);
    assertEquals(true, isFinded);
}
```

Рисунок 19 – Листинг кода метода testFindEmployeeById

На рисунке 20 приведен листинг кода метода testFindEmployeeByName который проверяет корректность работы метод findByName класса Employee.

```
@Test
public void testFindEmployeeByName() {
    Employee employee = new Employee();
    String name = "WrongName";
    boolean isFinded = employee.findByName(name);
    assertEquals(false, isFinded);
}
```

Рисунок 20 - Листинг кода метода testFindEmployeeByName

Проведя анализ существующих подходов к тестированию было решено проводить модульное тестирование, т.к. оно является наиболее подходящим для разработанной АСУ. Для написания тестовых классов на языке Java была выбрана библиотека JUnit поскольку является самой популярной библиотекой модульного тестирования программного обеспечения на используемом нами языке программирования.

Вывод по главе 2

В ходе проектирования автоматизированной системы управления рабочим пространством был выбран паттерн проектирования MVC. Данный шаблон оказался наиболее подходящим для обоснования архитектуры АСУ. Спроектировав диаграмму развертывания компонентов было описано

физическое оборудование, на котором будет выполнятся автоматизированная система управления рабочим пространством.

Все требования к системе были разбиты по группам на основе классификации требований FURPS+ и описаны в форме таблицы. Далее был проведен сравнительный анализ инструментов разработки ПО. В качестве языка программирования был выбран Java, а в качестве веб-сервера WildFly.

Для моделирования АСУ был выбран язык моделирования UML и программа Visual Paradigm, с их помощью были построены диаграмма вариантов использования системы и диаграмма классов. Так же в главе были приведены логическая и физическая модель хранения данных.

Были разработаны основные модули автоматизированной системы управления рабочим пространством и приведены примеры её работы.

После разработки завершающим этапом стало проведение тестирования разработанной системы. Были проанализированы основные подходы к тестированию и выбрано наиболее подходящее модульное тестирование. В качестве реализации данного подхода была использована библиотека JUnit.

Приведён пример теста одного из методов АСУ, для поиска сотрудника по имени или уникальному номеру, а также предоставлен листинг кода метода теста, с подробным описанием его работы.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была проанализирована литература по проблемам автоматизации основных бизнес-процессов деятельности компаний. После анализа литературы определились основные этапы автоматизации учета рабочего пространства.

Во время проведения анализа процесса учета рабочего пространства выяснилось, что процесс достаточно трудоёмкий и требует на свое выполнение большого количества ресурсов.

После разбиения процесса учета рабочего пространства на конкретные этапы, с использованием функциональной декомпозиции, стали видны конкретные этапы процесса, являющиеся наиболее трудоёмкими. Для этого была применена технология структурного анализа, на основе которого были построены диаграммы IDEF0.

Было принято решение в их автоматизации путём разработки автоматизированной системы управления рабочим пространством.

В ходе проделанной работы были изучены математические основы постановки задач динамического программирования. Сформирована математическая модель задачи оптимального распределения сотрудников по офису на основе динамического программирования. На основе сформированной математической модели был построен алгоритм её работы.

В дальнейшем были определены требования к системе по классификации требований FURPS+. Исходя из требований к системе и личного опыта был проведен анализ технологий для разработки программного обеспечения. В ходе анализа были построены сравнительные таблицы и выбраны соответствующие инструменты разработки.

После определения требований и анализа технологий разработки была построена архитектура разрабатываемого программного обеспечения. На графическом языке объектного моделирования UML была смоделирована диаграмма вариантов использования и диаграмма классов.

В результате работы была спроектирована реализована и протестирована информационная система для компании «Неткрэкер», осуществляющая автоматизацию деятельности отела кадров и администрации офиса, а именно процесс учета рабочего пространства. Задача оптимального распределения сотрудников по офису оказалась актуальной. Правильное управление рабочим пространством позволит повысить производительность труда и комфортность окружающей рабочей среды.

Автоматизировав некоторые этапы процесса учета рабочего пространства, сократится количество ресурсов и времени, затрачиваемое сотрудниками, работающими в отделе кадров и администрации офиса.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная методическая литература

1. PHP 7 / Д. В. Котеров, И. В. Симдянов — Спб.: БХВ-Петербург, 2016. — 1088 с.

2. Гагарина Л. Г., Киселев Д. В., Федотова Е. Л. Разработка и эксплуатация автоматизированных информационных систем М.: ИД «ФОРУМ», ИНФРА-М, 2012. - 384с.

3. Емельянова, Н.З. Проектирование информационных систем: учебное пособие / Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. - М.: Форум, 2014. - 432 с.

4. Информационные технологии в профессиональной деятельности : учеб. пособие / авт.-сост. А. А. Широких. - Пермь : Пермский гос. гуманит.-пед. ун-т, 2014. - 61 с.

5. Клочко И. А. Информационные технологии в профессиональной деятельности : учеб. пособие / И. А. Клочко. - Саратов : Вузовское образование, 2014. - 236 с.

6. Леонтьев, Б.К. Web-дизайн: тонкости, хитрости, секреты / Б.К. Леонтьев. – М.: Майор, 2013. – 176с.

7. Маклафлин, Б. PHP и MySQL : исчерпывающее руководство / Б. Маклафлин ; [пер. с англ. Н. Вильчинский]. – Санкт-Петербург : Питер, 2013. – 508 с.

8. Мишин А. В. Информационные технологии в профессиональной деятельности: учебное пособие / А. В. Мишин, Л. Е. Мистров, Д. В. Картавцев. - Москва : Российская академия правосудия, 2011. - 311 с.

9. Семахин А. М., Баталов И. С. Динамическое программирование в решении задачи оптимального размещения электронных компонентов системы управления // Молодой ученый. — 2013. — №6. — С. 144-146.

10. Силич В. А. Реинжиниринг бизнес-процессов : учеб. пособие / В. А. Силич, М. П. Силич. - Томск : ТУСУР, 2014. - 199 с.

11. Юдин К. А. Автоматизация проектирования с применением Autodesk Inventor 2012: учеб. пособие / К. А. Юдин ; Белгородский гос. технол. ун-т им. В. Г. Шухова. - Белгород : БГТУ : ЭБС АСВ, 2013. - 128 с.

Электронные ресурсы

12. "Кафедра телекоммуникационных сетей и систем" [Электронный ресурс]. – Режим доступа: http://iss.fizteh.ru/about.html

13. NEC and NetCracker Launch New Business Brand and Optimize NEC's Network Expertise and NetCracker's IT Leadership [Электронный ресурс]. – Режим доступа: http://www.nec.com/en/press/201502/global_20150224_01.html

14. NetCracker Announces Agreement to Acquire Subex's Activation Business [Электронный ресурс]. – Режим доступа: http://www.businesswire.com/news/home

15. Supporting the Profitable Operator of the Future [Электронный ресурс]. – Режим доступа: http://www.mobileeurope.co.uk/

16. Проектирование информационных систем // II Основы методологии проектирования информационных систем // п. 2.3. Содержание и организация проектирования / [Электронный ресурс]: http://webmath.exponenta.ru/db/02.html

Литература на иностранном языке

17. Dr. Danny Coward Java EE 7. The Big Picture / McGraw-Hill – 2015 – 513 p.

18. Farrell J. Java Programming/ Course Technology 2015 – 1026 p.

19. Oaks S. Java Performance: The Definitive Guide – O'Reilly, 2014.

20. Pilgrim, P. Digital Java EE 7 Web Application Development / P. Pilgrim — Packt Publishing, 2015. – 486 c.

21. Sam Newman, Building Microservices - O'Reilly Media, 2015. – 280 pages.

ПРИЛОЖЕНИЕ А

Листинг методов для нахождения оптимального рабочего места сотрудника

```
public void getMassiveOfPossibleWorkspaces() {
//Формирование множества возможных расположений
        for (int i = 0; i < Z.length; i++) {
            for (int j = 0; j < Z.length; j++) {
                Z[i][j] = "Employee: " + B[i] + " Workspace: "
+ M[j];
            }
        }
    }

public void getMassiveOfDistributions() {
//Заполнение массива критериев оптималости распределения
        for (int i = 0; i < B.length; i++) {
            for (int j = 0; j < M.length; j++) {
//Если отдел и проект совпадают то оптимальность = 1
                if(B[i].charAt(2)==M[j].charAt(4)            &&
B[i].charAt(4)==M[j].charAt(6)) {
                    C[i][j]=1;
                }
//Если совпадает только проект, то 0,8
                else  if  (B[i].charAt(2)==M[j].charAt(4)  &&
B[i].charAt(4)!=M[j].charAt(6)) {
                    C[i][j]=0.8;
                }
//Если совпадает только отдел, то 0,6
                else   if(B[i].charAt(2)!=M[j].charAt(4)   &&
B[i].charAt(4)==M[j].charAt(6)) {
                    C[i][j]=0.6;
                }
//Если совпадений нет, то 0,4
                else   if(B[i].charAt(2)!=M[j].charAt(4)   &&
B[i].charAt(4)!=M[j].charAt(6)) {
                    C[i][j]=0.4;
                }
//Если кабинет пустой, то 0,2
                else         if(M[j].charAt(2)=='0'         &&
M[j].charAt(6)=='0') {
                    C[i][j]=0.2;
```

```
                }
            }
        }
    }

public void getOptimalDistribution() {
//Выбор оптимальных распределений из массива имеющихся
        for (int i = 0; i < B.length; i++) {
            int count = 0;
            double max = 0.2;
            String[] optimalM = new String[M.length];
            for (int j = 0; j < M.length; j++) {
                if(C[i][j] > max){
                   max = C[i][j];
                   count=0;
                    for (int k = 0; k < optimalM.length; k++)
{
                       optimalM[k] = "";
                    }
                    optimalM[0]=M[j];
                }
                else if (C[i][j] == max) {
                    count++;
                    optimalM[count]=M[j];
                }
            }
        }
    }
```