

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРИТЕТ»
Институт Информационных технологий и управления в технических
системах
Кафедра Информационных технологий и компьютерных систем

Пояснительная записка

к выпускной квалификационной работе бакалавра
на тему: «Модель ERP-системы IT-фирмы»

Выполнил: студент 4-го курса, группы ИВТ/б-41-о
направления подготовки 09.03.01 — Информатика и вычислительная техника
профиль (специализация) 09.03.01.01 «ЭВМ, системы и сети»

Головня Александр Константинович

Руководитель: Чернышенко С.В., д.б.н., доцент

Дата допуска к защите «____» ____ июня _____ 2016 г.

Зав. кафедрой _____ А.А. Брюховецкий _____

2019 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
ПОСТАНОВКА ЗАДАЧИ	5
1 ОБЗОР СОВРЕМЕННЫХ ERP-СИСТЕМ НА РОССИЙСКОМ РЫНКЕ . 7	
1.1. Основные принципы ERP	7
1.2. Система модульности внедрения ERP-систем.....	8
1.3. Финансовый модуль ERP-систем.....	9
1.4. Модуль управления персоналом в ERP-системах	10
1.5. Операционный модуль ERP-систем.....	12
1.6. Обзорный анализ ведущих ERP-систем	13
1.6.1. SAP ERP	13
1.6.2. Oracle ERP.....	24
1.6.3. 1C ERP.....	31
2 АХИТЕКТУРА ПРОГРАММНОГО КОМПЛЕКСА	37
2.1. Подсистемы программного комплекса	37
2.2. Функции программного комплекса	46
2.3. Автоматизация инфраструктуры программного комплекса ...	47
3 ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ	49
3.1. Основные понятия	49
3.2. Применение веб-приложений.....	49
3.3. Структура веб-приложения.....	49
3.4. Популярные языки программирования для разработки	50
3.5. Доменные модели	51
3.6. Роль БД при создании веб-приложений	51

3.7. Популярные фреймворки для разработки веб-приложений...	52
3.7.1. Maven	52
3.7.2. Vaadin	53
3.7.2. Spring Framework	54
ЗАКЛЮЧЕНИЕ	55
ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ.....	56
БИБЛИОГРАФИЯ.....	57
ПРИЛОЖЕНИЕ А – Конфигурационный скрипт для стадии production.....	58
ПРИЛОЖЕНИЕ Б – Конфигурационный скрипт для стадии разработки.....	61
ПРИЛОЖЕНИЕ В – Листинг программы.....	62

ВВЕДЕНИЕ

Прохождение преддипломной практики обучающимися предусмотрено учебным планом подготовки бакалавров направления подготовки 09.03.01 «Информатика и вычислительная техника» в 8-м семестре в соответствии с положениями ФГОС от 12.01.2016 [1], Положением о практике обучающихся (студентов), осваивающих основные профессиональные образовательные программы высшего образования [2] и является завершающей практикой в период обучения студентов в университете.

В соответствии с рабочей программой практики [3] кафедры информационных технологий и компьютерных систем, основной целью практики является сбор и обработка материалов по теме выпускной квалификационной работы (ВКР) бакалавра, выполняемой непосредственно после практики, а также углубление и закрепление теоретических знаний, практических навыков и компетенций в процессе решения конкретных задач, связанных с этой темой.

В задачи практики входят:

(1) выполнение индивидуального задания по теме ВКР на период преддипломной практики (в дальнейшем – индивидуальное задание по теме ВКР);

(2) применение знаний из метрологии, стандартизации, нормоконтроля при составлении технической и программной документации.

ПОСТАНОВКА ЗАДАЧИ

Цель проектирования – разработка программно-аппаратного комплекса для моделирования ERP-системы IT-предприятия.

Модель ERP-системы IT-предприятия должна содержать следующие системные модули:

1. Модуль управления персоналом.
2. Модуль управления проектами.
3. Модуль продаж программных продуктов

В модуле управления персоналом должны быть предусмотрены возможности создания и удаления сотрудников, редактирования информации сотрудников, получения полного списка сотрудников с возможностью постраничного вывода и фильтрации по параметрам. Для каждого сотрудника должен быть предусмотрен свой личный кабинет с возможностью просмотра и редактирования личной информации. Для сотрудников должна быть предусмотрена ролевая система доступа, учитывающая их полномочия.

В модуле управления проектами должна быть предусмотрена возможность создания, редактирования и удаления проектов. Проект должен иметь заказчика и список сотрудников ответственных за выполнение этого проекта с распределением по ролям. Должно быть предусмотрена возможность получения отчета о текущем состоянии проекта и выполненных и задач и возможно отправлять подобный отчет по заданному расписанию клиенту. В интерфейсе необходимо предусмотреть возможность иерархического отображения проектов и просмотр текущих задач с фильтрацией по параметрам и сотрудникам.

В модуле продаж должна быть предусмотрена возможность создания внешним клиентом заявки на разработку программного продукта, отправки данной заявки ответственному сотруднику, автоматическому формированию первоначальной сметы и прайс-листа с оценкой трудозатрат. Также

необходимо предусмотреть возможность редактирования заявки и введения рабочих статусов для заявки. Итоговая подтвержденная заявка должна поступить в модуль управления проектами, где должен быть сформирован план работой, с назначенными сотрудниками и оцененными трудозатратами.

1 ОБЗОР СОВРЕМЕННЫХ ERP-СИСТЕМ НА РОССИЙСКОМ РЫНКЕ

ERP — это системная и организационная стратегия объединения различных направлений производственного процесса и его управления, такие как управление активами производства и финансовых операций, управление трудовыми ресурсами, организация финансового менеджмента, и при этом данный процесс ориентирован на постоянную балансировку и максимальную оптимизацию всех имеющихся ресурсов данного предприятия с помощью специально разработанного общего пакета программ прикладного программного обеспечения, которые могут создать и вывести общую модель данных и ведения всех необходимых процессов для всех направлений деятельности предприятия, на котором используется данная система.

ERP-система — это определенный программный пакет, который оптимизирует и помогает реализовать общую стратегию ERP.

1.1. Основные принципы ERP

Главная характеристическая черта ERP-стратегии – это возможность принципиального подхода к использованию единой модели транзакционной системы, которое можно применить для основного количества операций и всех текущих бизнес-процессов, идущих в организации. Причем данные системы могут быть применимы для любой функциональной и территориальной разобщённости процессов происходящих в производственном или ином процессе, независимо от причины их возникновения и происхождения, система даст возможность интегрировать сведения всех производимых операций в общую информационную базу для последующей системной обработки и получения результатов в реальном времени, а также выделения сбалансированных планов.

Отличительная черта ERP-системы – это возможность тиражирования. Это принцип дает возможность применения одного программного пакета для

любых предприятий и организаций, при этом для каждого из них возможно применить различные настройки и выставить необходимые расширения. Данная отличительная черта выступает как одно из главных условий внедрения ERP-систем. Так же еще одной из причин глобального применения тиражируемых ERP-систем, вместо индивидуальной разработки программного обеспечения, дает возможность применения лучших наработанных практик с помощью метода реинжиниринга бизнес-процессов согласно решениям, которые применяются в ERP-системе.

В связи с глобальным внедрением систем ERP, а также внедрением их в совершенно различные территориальные образования, в организации и предприятия, имеющие совершенно различные профили – требуется поддержка множества валют и языков, применимых в единой системе ПО. Кроме того, есть необходимость сопровождения многих организационных единиц единого процесса (это могут быть несколько юридических лиц, или же несколько предприятий одного холдинга или разобщенные поставщики одного производителя, или территориально удаленные филиалы одного холдинга), также использование нескольких планов счетов, причем принципиально разных, так же это могут быть различные схемы отчисления налогов, ведения бухгалтерского учета – все это необходимое условие использования ERP-систем в транснациональных холдингах и корпорациях.

1.2. Система модульности внедрения ERP-систем

Удобство внедрения ERP-систем в том, что их можно интегрировать в процесс поддержки производства поэтапно. Внедрять в эксплуатацию можно поочередно один или сразу несколько модулей с различной функциональностью. Причем этот процесс можно осуществлять на любом этапе деятельности, при этом устанавливать не все модули, а только те пакеты которые актуальны для организации или производства на сегодняшний день. Модульность использования ERP-систем дает возможность получать решения на основе использования сразу нескольких ERP-систем, при этом имея

возможность отбирать из каждой системы лучшее для себя. На сегодняшний день есть общепринятое для всех производителей разделение по модулям, а также их группировка – персонал, финансы, операции.

Уже начиная с 90-х годов в качестве модулей всех без исключения крупных ERP-систем вводились дополнения для систем клиентского обслуживания, возможностью управления персоналом, различными проектами, а также возможностью управления циклом производства. Но затем все данные модули начали поставляться как отдельно поставляемые информационные продукты в рамках ERP-систем, но при этом сохраняя основные требования преемственности в рамках действующих пакетов бизнес-приложений.

Универсальность и глобальная применимость ERP-систем в совершенно различных направлениях деятельности накладывает на них требование быть максимально универсальными, и при этом организовывать поддержку требований отраслевой специфики. Конечно, все крупные системы уже включают в свое программное обеспечение готовые модули и расширения «заточенные» для различных отраслей промышленности, и покупателю только остается для себя заказать уже готовый пакет дополнительного обновления. Среди таких пакетов можно выделить системы для предприятий добывающей промышленности, организаций сектора государственного управления, машиностроительных и обрабатывающих производств, розничной торговли, сферы образования и медицины, дистрибуции, финансовых организаций и банков, страховых компаний, предприятий электросвязи и энергетики, и многих других отраслей хозяйствования.

1.3. Финансовый модуль ERP-систем

Финансовые модули, вроде главной книги, компонента финансовой отчетности и due diligence (должностная добросовестность) — это основной компонент ERP-системы.

На сегодняшний день количество дополнительных финансовых модулей и блоков ERP огромно. Их можно систематизировать и выделить основные четыре направления. Это, прежде всего:

1. Бухгалтерский блок: главная книга, дебиторские счета, кредиторские счета, консолидированный бюджет.
2. Учетно-управленческий блок: счета учёт затрат и доходов предприятий и организаций, по учету продуктов производства или потребления, по выполняемым проектам, система калькуляции себестоимости выпускаемой или потребляемой продукции.
3. Казначейский блок: система управления ликвидностью предприятия и выпускаемой продукции, управление денежными средствами. В нее включены возможности контроля банковских счетов и управление кассой, система взаимодействия с банками, в которых расположены счета предприятия или организации и всех имеющихся подразделений и филиалов, управление кредитами и другими заимствованиями.
4. Финансово-управленческий блок: управление основными средствами производственного процесса, система управления инвестиционным менеджментом, управление финансовым контролем и управление возможными рисками предприятия.

По требованию заказчика в ERP-системы может быть включен модуль финансового планирования, а также управления основными показателями эффективности производства.

1.4. Модуль управления персоналом в ERP-системах

Главное отличие ERP, как стратегии развития предприятия или организации, от программ автоматизации определения дохода сотрудников – это объединение информации о трудовых ресурсах предприятия для эффективного планирования и управления всеми экономическими

операциями с учетом сведений о потенциальных возможностях задействованного персонала. Вторая отличительная черта – это возможность максимально точно определить и выявить затраты по мере их возникновения и объединить их с информацией о необходимых компенсациях задействованного в них работающего персонала.

Именно данный модуль ориентирует стратегию развития предприятия с учетом метода управления персоналом организации и предприятия как человеческим капиталом, и уже в рамках данной концепции есть возможность определить и внедрить функциональные особенности данных модулей. В них отображаются специфика управления персоналом, ведение информации о возможных профессиональных навыках каждого сотрудника, есть возможность планирования обучения в связи с изменениями производственного цикла, выстраивание карьеры и т.д. На основе всей данной информации, которая системно обрабатывается в этих модулях и выстраивается стратегическое управление всей организацией, происходит расчет финансового менеджмента, а также ключевых показателей эффективности.

Основными модулями управления персоналом являются:

1. Система подбора персонала;
2. Система кадрового учета;
3. Учет общего рабочего времени;
4. Система оплаты труда, выплатой премий;
5. Система управления разрядкой на выполнение работ;
6. Система выплат компенсаций и расчета заработной платы;
7. Система оценки персонала;
8. Организация расчетов производительности трудовых ресурсов предприятия;
9. Организация пенсионного учёта сотрудников;
10. Система управления повышения квалификации сотрудников.

1.5. Операционный модуль ERP-систем

Данные встраиваемые модули помогают скорректировать деятельность предприятий по созданию и реализации выпускаемых продуктов и предлагаемых услуг. Кроме того, в них есть все необходимые функции для оптимизации данных процессов. Несмотря на специфическую разобщенность различных сфер хозяйствования можно выделить несколько направлений операционных модулей:

1. Логистические: данные модули координируют снабжение, регулируют взаимоотношения с различными поставщиками, выстраивают управление всех поставок и транспортировкой товара, координируют складскую работу и управление запасами, отслеживают инвентаризацию основных средств производства;
2. Производственные: данные модули осуществляют производственное планирование, учёт выпускаемой и реализуемой продукции, системное управление всеми производственными программами данной организации;
3. Обеспечивающие: данные модули осуществляют управление технического обслуживания производственных комплексов, плановым и текущим ремонтом оборудования, планирование развития мощностей, управление транспортным потенциалом;
4. Сбытовые: данные модули координируют политику ценообразования, производят конфигурирование и обработку поступающих заказов, выстраивают систему продажи, продвижения товара и организацию послепродажного обслуживания.

1.6. Обзорный анализ ведущих ERP-систем

В данном разделе представлен анализ некоторых из имеющихся на отечественном рынке российских и западных систем, которые в той или иной степени можно отнести к ERP-системам.

1.6.1. SAP ERP

SAP — безусловный лидер по объемам продаж ПО данного класса в России. Компания держит порядка 40% всего российского рынка ERP-систем. Система R/3 относится к классу крупных интегрированных систем и имеет в своем составе модули, которые существенно расширяют рамки традиционной ERP-системы. Стоимость решения на 50 рабочих мест составляет ориентировочно около \$350 тысяч. Стоимость внедрения как минимум равна стоимости лицензий, а чаще всего в несколько раз превышает ее. Срок внедрения зависит от требуемых функциональных возможностей. Можно сказать, что для российских предприятий он в среднем составляет год-два. Один из наиболее полномасштабных проектов внедрения системы R/3 осуществлен на Омском нефтеперерабатывающем заводе.

SAP ERP является особой информационной системой ERP, которая обеспечивает полное планирование всех ресурсов предприятия. Данная информационная оболочка предназначена для полной автоматизации всех видов деятельности компании.

Компания SAP сегодня занята разработкой и внедрением автоматизированных систем, которые позволяют управлять всеми внутренними процессами предприятия. Среди них можно выделить:

1. Система анализа и контроля бухгалтерского учета;
2. Система анализа и контроля торговли предприятия;
3. Система анализа и контроля производственного цикла;
4. Система анализа и контроля финансовой деятельности;
5. Система анализа и контроля управлением персонала;

6. Система анализа и контроля управление складами, ревизорская деятельность;

Приложения без труда можно адаптировать под законодательную базу любой страны. Помимо продажи программного обеспечения, компания SAP предлагает многочисленные и квалифицированные услуги по внедрению его в реальном секторе экономики.

Сегодня, основная ERP-система компании SAP, официально носит название – SAP ERP ECC (Enterprise Core Component). Возможности последнего поколения SAP ERP дает возможность охватить все сферы деятельности предприятия. Среди данных направлений можно выделить: финансовый и управленческий учета, автоматизированную систему управления персоналом, информационный модуль, отвечающий за оперативную деятельность предприятия, а также аналитические справки о деятельности корпоративных сервисных служб. Но, основным направлением применимости данного ПО, можно считать формирование и предоставление подробных аналитических справок. Для формирования, которых внедрены специальные инструменты.

Система SAP ERP полностью состоит из набора модулей, которые можно интегрировать в один общий пакет и которые поддерживают практически все бизнес-процессы, идущие в производственном ли ином цикле, при этом все модули интегрированы друг с другом и могут обмениваться информацией в режиме реального времени.

Модуль «Транзакция SAP» – это прикладная программа, которая выполняет определенный бизнес-процесс в системе управления предприятием (это может быть совершение проводки денежных средств по расчетным счетам, или проводка счета-фактуры, формирование некоторого отчета и т.д.) Данный модуль осуществляет оперативный надзор над данными, и совершает логически завершенный и определенный набор действий.

Вся Система разбита на отдельные модули, при этом каждый модуль также состоит из определенного количества транзакций, которые должны охватывать определенную часть функционирования предприятия. Границы модулей, по сути, очень условны, между ними непрерывно происходит обмен данными, кроме того, отдельные программные модули вполне могут иметь общие настройки, рабочие таблицы с интегрированными данными.

Модуль – Финансы (FI)

Эта часть программного обеспечения предназначена для организации бухгалтерской отчетности предприятия организации или иной формы деятельности. Он включает в себя:

1. Функции формирования отчетности по дебиторам, кредиторам и вспомогательной бухгалтерии;
2. Функции формирования отчетности и занесения в Главную книгу (гроссбух);
3. Функции формирования отчетности «Бухгалтерию дебиторов»;
4. Функции формирования отчетности «Бухгалтерию кредиторов»;
5. Функции формирования отчетности «Финансовое управление»;
6. Функции формирования отчетности «Специальный регистр»;
7. Функции формирования отчетности «Консолидация»;
8. Интегрированную информационную систему учета и отчетности по финансовой деятельности.

Модуль Контроллинг (CO)

Этот Модуль дает возможность вести учет затрат и прибыли предприятия в целом, так и по каждому отдельному звену производственного цикла. Он включает в себя:

1. Возможность формирования отчета «Учет затрат по местам их возникновения»
2. Возможность формирования отчета «Учет затрат по заказам»;

3. Возможность формирования отчета «Учет затрат по проектам»;
4. Провести «Калькуляцию затрат»;
5. Провести «Контроль прибыльности (результатов)»;
6. Возможность формирования отчета «Контроль мест возникновения прибыли (центров прибыли)»;
7. Возможность формирования отчета «Учет выработки, Контроллинг деятельности предприятия».

Модуль – Управление основными средствами (AM)

Фактически этот модуль необходим для учета основных средств предприятия и методами управления ими. Основные элементы данного модуля:

1. Блок «Техническое управление основными средствами производства»;
2. Блок «Техобслуживание и ремонт оборудования производства»;
3. Блок «Контроллинг инвестиций и продажа активов»;
4. Блок «Традиционный бухучет основных средств»;
5. Блок «Замена основных средств и амортизация оборудования и основных средств производства»;
6. Блок «Управление инвестициями компании».

Модуль – Управление проектами (PS)

Данный модуль имеет прикладную направленность. Модуль PS поддерживает структурное планирование, управление всеми циклами производства, отслеживание и координацию долгосрочных проектов с любым уровнем сложности. Основные элементы модуля PS:

1. Возможность координации направления «Контроль финансовых средств и ресурсов»;
2. Возможность координации направления «Контроль качества»;

3. Возможность координации направления «Управление временными данными»;
4. Информационная система управления проектами,

Модуль – Производственное планирование (PP)

Этот Модуль в основном используется для организации долгосрочного планирования и выставления функций контроля всей деятельности предприятия в целом. Основные элементы данного модуля:

1. Производственные заказы
2. Технологические карты
3. Спецификации (BOM)
4. Планирование потребности в материалах (MRP)
5. Калькуляция затрат на изделие
6. Рабочие центры (места)
7. Планирование непрерывного производства
8. Планирование сбыта (SOP)
9. Производственное планирование (MPS)
10. Управление производством (SFC)
11. Канбан (Just in time)
12. Учет затрат по процессам
13. Серийное производство

Модуль – Управление материальными потоками (MM)

Данный модуль поддерживает операции снабжения и управления запасами в организации деятельности предприятия, а также данный модуль применим в различных хозяйственных операциях проводимыми предприятием. Основные элементы модуля:

1. Организация приобретения материалов;
2. Организация управление запасами;
3. Организация управление складами;

4. Систематизация контроля счетов предприятия;
5. Организация оценки запасов необходимых материалов;
6. Организация аттестация услуг и товаров поставщика;
7. Обработка данных по выполняемым работам и услугам;
8. Создание базы информационной системы управления запасами предприятия;

Модуль – Сбыт (SD)

Данный модуль очень важен он вносит ясность в политику реализации конечного продукта предприятия, помимо этого он решает задачи распределения конечного продукта, организацию продаж, определяет систематику поставок и конечного выставления счетов. Основные элементы модуля:

1. Организация предпродажная поддержки на производстве;
2. Возможность формирования отчета «Обработка запросов»;
3. Возможность формирования отчета «Обработка предложений»;
4. Возможность формирования отчета «Обработка заказов»;
5. Возможность формирования отчета «Обработка поставок»;
6. Организация выставление счетов (фактурирование);
7. Блок «Информационная система сбыта».

Модуль – Управление качеством (QM)

Этот модуль интегрирует в себя всю информационную систему компании, а также контролирует систему управления качеством. Кроме этого в него встроены функции которые обеспечивает мероприятия направленные на планирования качества товаров и услуг данной компании, осуществление проверки и контроля качества продукции на всех этапах ее производства, а также при ее закупках. Основные элементы модуля:

1. осуществление проверки качества;
2. организация планирования качества;

3. информационная поддержка контроля качества выпускаемой продукции(QMIS).

Модуль Техобслуживание и ремонт оборудования предприятия (PM)

Данный модуль незаменим в процессе учета затрат и на этапе планирования расхода ресурсов на выполнение, текущего техобслуживание и плановый ремонт основных средств производства. Основные элементы модуля:

1. формирование запроса «Незапланированный ремонт»;
2. формирование запроса «Управление сервисом»;
3. формирование запроса «Планово-профилактический ремонт»;
4. создание отчета «Ведение спецификаций»;
5. организация информационной системы технического обслуживания и ремонта основных средств производства.

Модуль – Управление персоналом (HR)

Это полностью интегрированная система, которая предназначена для планирования и управления работой всего персонала, задействованного в цикле деятельности компании. Основные элементы модуля:

1. Администрирование деятельности персонала;
2. Анализ и Расчет зарплаты сотрудников;
3. Система управление временными данными персонала;
4. Система расчета расходов сотрудников на командировки;
5. Определение льгот;
6. Система приглашения и набора нового персонала;
7. Организация работ по повышение квалификации работающего персонала;
8. Организация процесса оптимального использования рабочей силы предприятия;

9. Организация и проведения семинаров и обучающих мероприятий;
10. Организационный и тайм-менеджмент;
11. Блок обработки информации о систематике персонала.

Модуль – Управление информационными потоками (WF)

Данный модуль как интегрированная единица в своей роли связывает прикладные модули с встроенными технологиями ERP-системы, также всеми сервисными средствами и инструментами данного информационного продукта. Возможность управления всем потоком операций с возможностью автоматизированного контроля всех хозяйственных процессов по заранее внесенному алгоритму анализа по заранее определенным и прописанным процедурам и правилам. Кроме того, данный модуль имеет офисную систему со своей встроенной электронной почтой, также систему управления документооборотом компании, загруженный универсальный классификатор, а также возможность произвести интеграцию с любой САПР. Если в системе происходит какое-то определенное событие, то одновременно с ним запускается протокол этого события и включается соответствующий процесс. Модуль включает диспетчер потока выполняемых системой операций и при этом инициирует входящую единицу потока операций (Workflow Item). Затем система объединяет входящие данные, затем документы объединяются, и происходит процесс обработки информации в соответствии с определенной встроенной логической схемой.

Модуль – Отраслевые решения (IS)

Данный модуль интегрирует встроенные прикладные модули SAP, SAP R/3, а также и дополнительные специфические программы конкретно для каждой отрасли хозяйствования. На сегодняшний день разработаны и с легкостью могут быть интегрированы в единый модульный пакет отраслевые решения сопровождения бизнеса:

1. Пакет отраслевых приложений «авиация и космос»;
2. Пакет отраслевых приложений «оборонная промышленность»;
3. Пакет отраслевых приложений «автомобильная промышленность»;
4. Пакет отраслевых приложений «нефтяная и газовая промышленность»;
5. Пакет отраслевых приложений «химическая промышленность»;
6. Пакет отраслевых приложений «фармацевтическая промышленность»;
7. Пакет отраслевых приложений «машиностроительная промышленность»;
8. Пакет отраслевых приложений «товары народного потребления»;
9. электронная и непроизводственной сферы:
10. Пакет отраслевых приложений «банковское дело»;
11. Пакет отраслевых приложений «страхование»;
12. Пакет отраслевых приложений «государственное и муниципальное управление»;
13. Пакет отраслевых приложений «телекоммуникационные технологии»
14. Пакет отраслевых приложений «коммунальное хозяйство»;
15. Пакет отраслевых приложений «здравоохранение»;
16. Пакет отраслевых приложений «розничная торговля».

Модуль – Базисная система

Данный модуль служит основой для информационной системы SAP R/3. Он должным образом гарантирует полноценную интеграцию всех прикладных модулей и полную независимость от аппаратной платформы, на которую установлено данное программное обеспечение. Также базисная система дает возможность организации работы в системе многоуровневого распределения архитектуры – «клиент-сервер».

Базис – это особый модуль. Его функционал значительно шире изложенной информации. От его функционирования зависит работоспособность системы в целом. Администраторы базисного модуля несут полную ответственность за общее функционирование SAP. Задачи базисного модуля:

1. Начальное прописывание всех установок и настройка всех встроенных параметров производительности системы в целом;
2. Выстраивание системы администрирования всех встроенных баз данных;
3. По мере необходимости обновление ПО системы и установка необходимых пакетов обновлений модулей и корректур;
4. Организация и осуществление переносов в продуктивную систему;
5. Основное администрирование проекта – главный ввод и присвоение всех ролей пользователям, участвующим в организации работы по данному проекту;
6. Организация процесса резервного копирования промежуточных и финальных данных по проводимым операциям;
7. Основная настройка взаимодействия отдельных систем участвующих в процессе анализа и обработки данных;
8. Организация контроля системы, с прописанием задачи ПО – заблаговременно определять и выявлять возникающие проблемы и принять все необходимых мер по их устранению;
9. Организация настройки доступа к интегрированным модулям и системам службам поддержки SAP;
10. Анализ выдаваемых ошибок и их устранение;

Сегодня система SAP ERP является самой обширной программной оболочкой среди подобных себе информационных пакетов. Потому практически все лидеры мировой экономики выбрали ее в качестве своей корпоративной системы управления производством. При этом, по данным

статистики, что примерно 30% всех компаний, приобретающих систему SAP R/3 – это вовсе не гиганты экономики, а фирмы с оборотом средств менее 200 миллионов долларов в год.

1.6.2. Oracle ERP

Позиции компании Oracle в России существенно слабее, чем у ее основного конкурента. Стоимость решения на базе Oracle Applications несколько ниже, чем на базе R/3 (конкретных цифр в открытой печати не приводилось). Срок внедрения у Oracle Applications и R/3 примерно одинаков. Из наиболее известных проектов внедрения Oracle Applications можно отметить реализованный на Магнитогорском металлургическом комбинате.

Компания Oracle пошла по пути разработки различных модулей, предназначенных для решения узконаправленных задач. Множество модульных систем Oracle объединяются в определенные бизнес пакеты, которые в свою очередь в дальнейшем интегрируются и «дописываются» под нужды заказчика.

Для решения потребностей ERP систем был создан модуль «Oracle E-Business Suite». Системой, разработанной на Oracle, пользуются такие учреждения как МВД РФ, ФСБ РФ, Сбербанк России, Центральный Банк РФ (ЦБ РФ), ФСНП (Федеральная служба налоговой полиции), Билайн, Промстройбанк, Comstar, Банк Москвы, и многие другие. Oracle занимает около 18% рынка интегрированных корпоративных систем.

Oracle E-Business Suite (OEBS) раньше имело название Oracle Applications. OEBS одно из немногих решений, которое включает в себя весь необходимый функционал для управления логистикой, сбытом и продажами, маркетингом, обслуживанием клиентов и заказчиков, персоналом (HR), производством, финансами, взаимодействием с поставщиками, и многими другими модулями.

Oracle E-Business Suite прекрасно интегрируется и с другими решениями компании Oracle, за счёт чего можно очень быстро расширить функционал ERP-системы внутри компании, таким образом ваша компания получает

мобильность и независимость при расширении производства. Отличительная особенность Oracle – это полный охват как конкретных жизненных циклов и процессов на самом низком уровне действий, так и мощная система отчетности для директоров и управляющих, что позволяет видеть картину бизнеса полностью. При внедрении Oracle ERP на территории заказчика необходимо определить действующую корпоративную информационную систему, определить пути развития информационной системы, необходимо составить полную карту бизнес-процессов, нужно составить бизнес требования со стороны заказчика к информационной системе и определить, в каких местах корпоративная информационная система может автоматизировать процессы.

Основные модульные сектора Oracle E-Business Suite:

1. Управление производством
2. Финансы
3. Управление жизненным циклом
4. Управление логистикой
5. Управление проектами
6. Управление техобслуживания и ремонта
7. Управление эффективностью бизнеса (CPM)
8. Управление материальными потоками
9. Управление взаимоотношениями с клиентами
10. Система управления персоналом
11. Финансовый сервис

Управление отношениями и взаимодействием с клиентской базой

За управление отношениями отвечает модуль Customer Relationship Management (CRM), в него входят следующие решения:

1. Oracle Channel Revenue Management
2. Oracle Marketing
3. Oracle Order Management
4. Oracle Service

Управление услугами

Для управления услугами существует решение Service, которые отвечает за информационное обслуживание клиентов как по средствам телефонной связи, email, контакт-центра, «умной поддержки», и др. В него входят следующие решения:

1. Advanced Inbound Telephony
2. Advanced Outbound Telephony
3. Advanced Scheduler
4. Spares Management
5. TeleService
6. Depot Repair
7. Interaction Center
8. iSupport
9. Mobile Field Service
10. Scripting
11. Service Contracts
12. Email Center
13. Field Service

Управление финансами

Это один из наиболее интересных модулей системы OEBS. Oracle E-Business Suite Financials отвечает полностью за финансовую часть вашей компании, беря полностью на себя весь денежный оборот внутри и во вне фирмы (финансовая аналитика, отчеты, кредитование, зарплаты, управление активами, управление «treasury» или ценными предметами, финансовый жизненный цикл активов, и др). Следующие модули входят в состав Oracle E-Business Suite Financials:

1. Financial Control & Reporting
2. Asset Lifecycle Management
3. Procure-To-Pay
4. Cash & Treasury Management
5. Governance, Risk and Compliance
6. Credit-To-Cash
7. Financial Analytics
8. Lease and Finance Management
9. Travel and Expense Management

Управление человеческими активами или управление человеческим капиталом (Human Capital Management)

Модуль HCM включает в себя решения, позволяющие наладить контакт внутри фирмы, так называемый Team Building. Здесь есть и модули для службы персонала (отдела кадров), и управление человеческими ресурсами, создание отчетов, моделирование процессов нагрузки на человеческий ресурс, и Talent Management. Следующие программы входят в состав HCM:

1. Workforce Service Delivery
2. Global Core Human Capital Management
3. Приложения Talent Management
4. Workforce Management

5. HR Analytics

Управление проектами (Project Portfolio Management)

Данное решение позволяет полностью управлять проектами, взаимодействием внутри компании по решению проектов, назначать ответственных лиц, строить отчеты и аналитику по успешности проекта, управлять закупками товаров/материалов в рамках проекта, здесь есть мониторинг и составление документации проектов. Полный список приложений внутри PPM:

1. iProcurement
2. Supplier Lifecycle Management
3. Oracle Contract Lifecycle Management for Public Sector
4. iSupplier Portal
5. Oracle Procurement & Spend Analytics
6. Services Procurement
7. Sourcing
8. Oracle Spend Classification
9. Oracle Supplier Network
10. Procurement Contracts
11. Purchasing
12. Oracle Supplier Hub
13. Landed Cost Management

Управление цепочками поставок

Данное решение интегрирует в компанию модули, которые позволяют наладить цепочку поставок и процессов поставки, процессы управления логистическими шагами, оптимизировать планирование и закупки. В состав SCM (Supply Chain Management) входят следующие решения Oracle:

1. Advanced Procurement
2. Business Intelligence and Analytics

3. Value Chain Execution
4. Value Chain Planning
5. Order Orchestration and Fulfillment
6. Manufacturing
7. Asset Lifecycle Management
8. Product Value Chain Management

Планирование цепочки создания стоимости

Данное решение Value Chain Planning предназначается для оптимизации бизнес-процессов, с целью уменьшения стоимости конечного продукта, либо для снижения издержек на производство. VCP прекрасно интегрируется с другими решениями, а также с JD Edwards EnterpriseOne. В состав Value Chain Planning входят следующие модули:

1. Advanced Planning Command Center
2. Value Chain Planning для клиентов JD Edwards EnterpriseOne (PDF)
3. Advanced Supply Chain Planning
4. Strategic Network Optimization
5. Collaborative Planning
6. Service Parts Planning
7. Demand Management
8. Real-Time Sales and Operations Planning
9. Demand Signal Repository
10. Global Order Promising
11. Rapid Planning
12. Inventory Optimization
13. Production Scheduling
14. Predictive Trade Planning and Optimization

Создание стоимости

Value Chain Execution – это дополнительное решение, схожее с планированием цепочки создания стоимости, однако отличающееся от него программной составляющей. VCE (Value Chain Execution) позволяет управлять складскими запасами, транспортировкой, мобильностью компании, учетом инвентаря. В состав Value Chain Execution входят следующие программные решения:

1. Transportation Management
2. Inventory Management
3. Landed Cost Management
4. Mobile Supply Chain
5. Global Trade Management
6. Warehouse Management

1.6.3. 1C ERP

Несмотря на то, что у компании 1С есть решения УПП и 1С:Предприятие, они не решают полностью задач ERP. Однако стоимость лицензии и технические аппаратные средства значительно дешевле, нежели у конкурентов из Oracle или SAP. При этом программный код 1С осваивается быстрее и более понятен в отечественной специфике, что позволяет значительно быстрее интегрировать различные решения 1С внутри фирмы. При этом недостаточный функционал УПП или 1С:Предприятия восполняется множеством других программ, которые могут формировать один кластер ERP.

В управлении производством посредством программного оборудования заложены два уровня. Первый уровень, является уровнем логиста, то есть главного диспетчера предприятия. Второй уровень является цеховым, то есть локальным уровнем управления.

На первом уровне, где планирование осуществляет главный диспетчер, происходит разработка графика производства. Все производственные заказы выставляются в очередь, согласно приоритетности и сроков исполнения. Затем они вносятся в график производства, в котором, в свою очередь учитывается доступность мощности производства и доступность, необходимых для конкретного заказа материальных ресурсов. После этого, каждый заказ проходит деление на этапы и интервалы планирования. Затем каждый из интервалов закрепляется за отдельным исполняющим подразделением, которое будет работать над данным заказом.

На втором уровне управления производственными процессами, график производства отслеживается цеховыми диспетчерами, цель работы, которых в выделенном подразделении, создать расписание для рабочего центра. Также локальные диспетчеры управляют отклонениями. Подразделения имеют право выбирать одну, наиболее приемлемую модель управления. Таким образом, в зависимости от неё, расписание может быть составлено для всех центров, которые обеспечивают работу предприятия или для узких мест процесса

производства, согласно методологии ТОС (когда обеспечивается поддержание варианта, при котором автоматически определяется ряд рабочих центров, через определение вида суммарной нагрузки на оборудование). В третьем варианте работы расписание вообще может не составляться, тогда действует упрощенная схема расчета нагрузки на оборудование и управление производится для общей длительности этапа производства.

На отдельных этапах производства производителя контроль исполнения нормативов посредством маршрутных листов.

На производстве вводится “семафорная” система для оповещения. Она позволяет диспетчеру определить зону контроля производством, что снизит трудозатраты. Будут выявлены неблагоприятные и проблемные участки производства. Таким образом этот инструмент позволяет специалистам, обеспечивающим контроль производственным процессом составить прогноз неблагоприятного развития ситуации. Таким образом, будет сокращено количество неприятных ситуаций, касающихся запаздывания производства, задержки партий выпускаемой продукции, и срывов выпуска.

Обслуживание и ремонт оборудования также требует автоматизации управления.

Все объекты эксплуатации делятся на определенные классы, в зависимости от специфических характеристик, близости состава, паспортных данных, показателей наработки, необходимости ремонтных работ, близости режима эксплуатации. Учитывается состояние, в котором находится объект, его расположение в определенный период времени и принадлежность. Обслуживание этих объектов может осуществляться с детализацией до ремонтного узла.

Вводимая система дает возможность постоянного мониторинга объектов, с учетом их состояния, выявления дефектов и наработок, что дает

возможность заблаговременного планирования мероприятий, направленных на надлежащий и своевременный ремонт оборудования.

Также возможно привязывание контролируемых объектов эксплуатации к рабочему цеху производства. В этом случае ремонтные работы необходимо учитывать при производственном планировании, так как на время проведения ремонтных работ отдельные рабочие центры становятся недоступны для производственных целей. В свою очередь к проведению ремонта возможно привлечение любых производственных ресурсов. Причем производство может быть привлечено к удовлетворению нужд персонала, привлеченного к ремонту оборудования.

В итоге, очевидно, что при объединении подсистемы, занимающейся управлением ремонтными работами и производственной подсистемы, пользователь получает возможность формировать единую систему обеспечения потребностей предприятия. Причем в ту систему будет входить деятельность всего предприятия, на основании которой будет выведена конечная стоимость обслуживания объектов эксплуатации.

Проведение мониторинга и анализа показателей деятельности

Что позволяет привнести в эту область деятельности предприятия новое решение? В нем заложены уникальные механизмы, которые позволят с легкостью строить иерархию целей и показателей, проводить мониторинги отдельных показателей, причем расшифровывая исходные данные, а также анализировать финансовые результаты в каждом из направлений деятельности предприятия.

Управления финансовыми потоками

Для финансистов крупных предприятий программа также дает новые возможности. В систему введена возможность ведения табличного ввода данных, а также их последующей коррекции, причем с сохранением истории вносимых изменений. Внесены разнообразные средства, которые позволяют

автоматически рассчитывать все статьи бюджета, а также расшифровывать их до исходных значений. Причем в каждой из статей использовать до 6 уровней аналитики. Теперь есть возможность использования не одного, а нескольких источников при расчете каждого из показателей. Рассчитывать их можно в экранной версии ведения и редакции бюджета. Такая усовершенствованная структура дает возможность формирования финансовых прогнозов и анализа выполнения плановых показателей.

Программное решение повышает функциональные возможности ведения учета всех операций компании, в частности выдачи кредитов и займов, прозрачными становятся операции, проводимые с использованием пластиковых карт. Также возможно ведение платежного календаря и установление полного контроля над проводимыми операциями. Внесены в систему инструменты, позволяющие формировать платежные соглашения на будущие даты, проводить согласование расходования денежных активов, и проводить плановую инвентаризацию расчетных счетов предприятия и касс.

Программа позволяет вести отдельный управленческий и регламентированный учет. Также осуществлять контроль за лимитами задолженностей, причем он ведется в автоматическом режиме. Облегчается проведение инвентаризации, в том числе и относительно взаиморасчетов. В итоге своей деятельности эта часть программы выводит несколько видов отчетности, в частности статику и анализ по состоянию взаиморасчетов со всеми контактирующими предприятиями.

Относительно проведения деятельности в рамках регламентированного учета, нужно отметить, что решение позволяет без дополнительных затрат времени и сил вести автоматизированный учет. При осуществлении отчетности, используется версия единого плана счетов. Правила, которые лежат в основе отражения операций компании настраиваются самостоятельно. Те факты хозяйственного учета, которые отражаются в блоке оперативного учета, подвергаются детализации по значимости и актуальности в

соответствии с первичными документами, а затем фиксируются в регламентированном учете. Различные формы отчетности автоматически обновляются с помощью интернет ресурсов. Также возможен учет деятельности тех предприятий, которые вывели отдельные из своих подразделений на самостоятельный баланс.

Программа снабжена настроенной методической моделью, которая позволяет без лишних усилий вести отчетность по МСФО. В неё входят и шаблоны проводок, и планы счетов, и финансовые отчеты. Дается возможность отражать проводки в учете, создавать отдельные документы по типовым операциям, регистрировать финансовые и нефинансовые показатели.

Благодаря новейшим программным разработкам, возможно, поддержания определенной структуры склада, иерархии. Даже на больших складах возможно проведение системной инвентаризации, при этом не будет необходимости в прекращении их работы даже на короткий период времени. Есть возможность организации мобильных рабочих мест для работников склада. Дополнительная возможность резервировать материальные ценности в рамках заказов.

Относительно проведения закупок, то отмечается, что возможен выбор поставщиков по результатам глубокого анализа предлагаемых условий сотрудничества. Также можно контролировать возникающие потребности и качество их удовлетворения.

Есть возможность повысить уровень продаж, благодаря анализу проводимых мероприятий, грамотному формированию цен и прайс-листов.

Также есть функция проведения постоянного мониторинга уровня и состава продаж и заказов клиентов.

Пользователь в отношении с клиентами может вести досье каждого постоянного клиента, вводить карты лояльности. А также осуществлять постоянный анализ работы менеджеров и торговых представителей.

Что касается расчета себестоимости выпускаемой продукции, то осуществляется постоянный контроль над количеством потраченных ресурсов, на основе данных оперативного учета. Оценка себестоимости проводится в нескольких валютах, которые изначально устанавливаются пользователем. Учитываются затраты относительно всех видов деятельности.

Предлагаемое программное обеспечение имеет ряд важных технологических достоинств. Оно обеспечивает пользователю надежность, масштабность и производительность систем, организацию работы с персоналом и клиентами в режиме реального времени, использование возможности входа в систему с помощью устройств связи, работающих на Android, возможность настройки индивидуализированного интерфейса. Пользователь может включать отдельные части решения без изменения конфигурации.

2 АХИТЕКТУРА ПРОГРАММНОГО КОМПЛЕКСА

2.1. Подсистемы программного комплекса

Всем Spring Boot приложениям в этом проекте для старта необходим доступный Config Server. Благодаря опции fail-fast в bootstrap.yml каждого приложения и опции restart: always в докере, контейнеры можно запускать одновременно (они будут автоматически продолжать попытки старта, пока не поднимется Config Server).

Механизм Service Discovery так же требует некоторого времени для начала полноценной работы. Сервис не доступен для вызова, пока он сам, Eureka и клиент не имеют одну и ту же мета-информацию у себя локально — на это требуется 3 хартбита. По умолчанию период времени между хартбитами составляет 30 секунд.

Код программы основных контроллеров микросервисных приложений приведен в Приложении В.

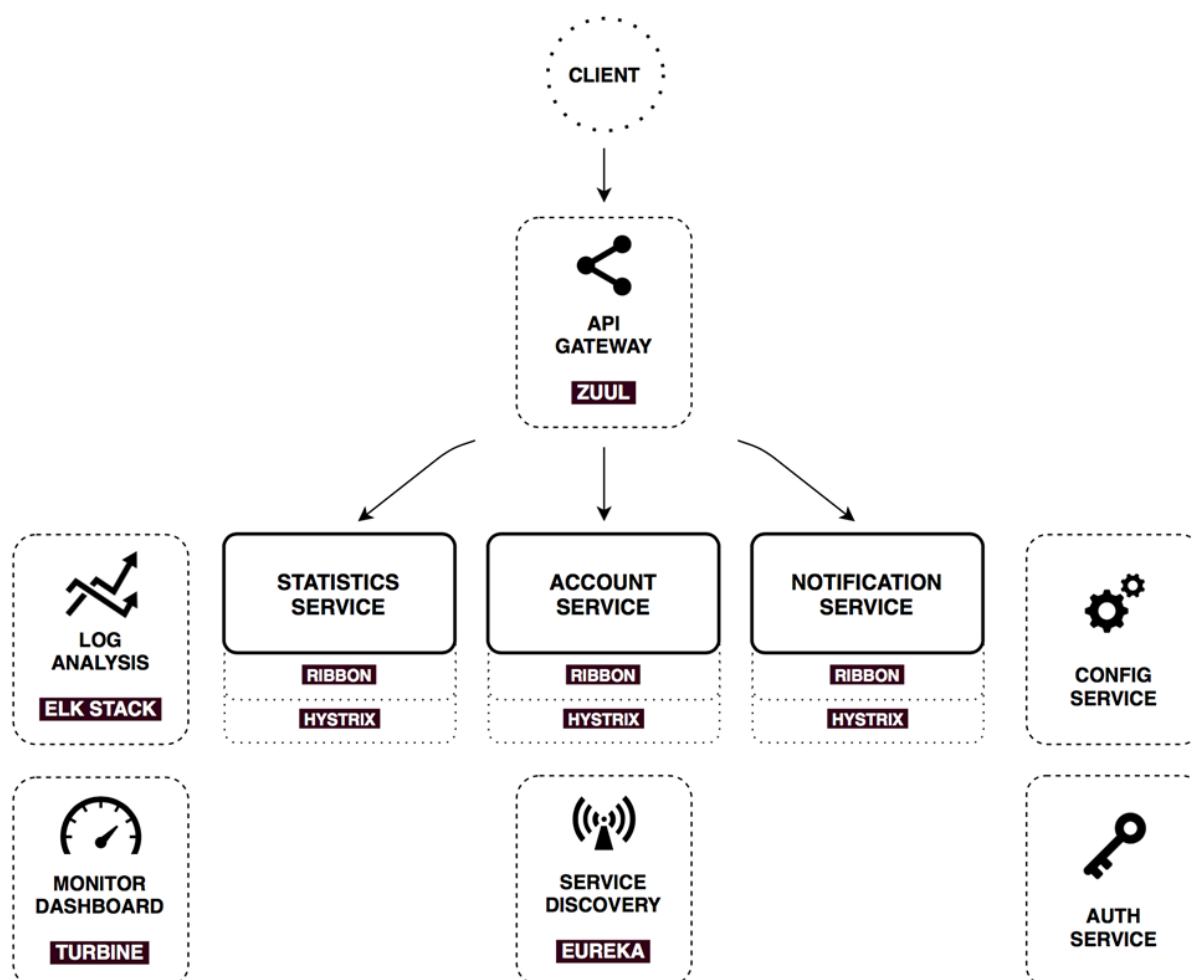


Рисунок 1 – Архитектура программно-аппаратного комплекса

Config Server

Spring Cloud Config — это горизонтально масштабируемое хранилище конфигураций для распределенной системы. В качестве источника данных на данный момент поддерживаются Git, Subversion и простые файлы, хранящиеся локально. По умолчанию Spring Cloud Config отдает файлы, соответствующие имени запрашивающего Spring приложения (но можно забирать свойства под конкретный Spring profile и из определенной ветки системы контроля версий).

На практике наибольший интерес представляет загрузка конфигураций из систем контроля версий, но здесь для простоты будем использовать

локальные файлы. Поместим директорию `shared` в класспате приложения, в которой будут храниться конфигурационные файлы для всех приложений в кластере. Например, если Notification service запросит конфигурацию, Config server ответит ему содержимым файла `shared/notification-service.yml`, смерженным с `shared/application.yml` (который является общим для всех).

На стороне клиентского приложения теперь не требуются никаких конфигурационных файлов, кроме `bootstrap.yml` с именем приложения и адресом Config server:

```
spring:
  application:
    name: notification-service
  cloud:
    config:
      uri: http://config:8888
      fail-fast: true
```

Auth Server

Обязанности по авторизации полностью вынесены в отдельное приложение, которое выдает OAuth2 токены для доступа к ресурсам бэкенда. Auth server используется как для авторизации пользователей, так и для защищенного общения сервис-сервис внутри периметра.

На самом деле, здесь описан только один из возможных подходов. Spring Cloud и Spring Security позволяют достаточно гибко настраивать конфигурацию под ваши нужды (например, имеет смысл проводить авторизацию на стороне API Gateway, а внутрь инфраструктуры передавать запрос с уже заполненными данными пользователя).

В этом проекте я использую Password credential grant type для авторизации пользователей и Client credentials grant type — для авторизации между сервисами.

Spring Cloud Security предоставляет удобные аннотации и автоконфигурацию, что позволяет достаточно просто реализовать описанный функционал как со стороны клиента, так и со стороны авторизационного сервера.

Со стороны клиента это ничем не отличается от традиционной авторизации с помощью сессий. Из запроса можно получить объект `Principal`, проверить роли и другие параметры с использованием аннотации `@PreAuthorize`.

Кроме того, каждое OAuth2-приложение имеет `scope`: для бэкенд-сервисов — `server`, для браузера — `ui`. Так мы можем ограничить доступ к некоторым эндпоинтам извне:

```
@PreAuthorize("#oauth2.hasScope('server')")
@RequestMapping(value = "accounts/{name}", method = RequestMethod.GET)
public List<DataPoint> getStatisticsByAccountName(@PathVariable String name) {
    return statisticsService.findByAccountName(name);
}
```

API Gateway

Гипотетически, клиентское приложение могло бы запрашивать каждый из сервисов самостоятельно. Но такой подход сразу натывается на массу ограничений — необходимость знать адрес каждого эндпоинта, делать запрос за каждым куском информации отдельно и самостоятельно мерджить результат. Кроме того, не все приложения не бэкенде могут поддерживать дружественные вебу протоколы, и прочее прочее.

Для решения такого рода проблем применяют API Gateway — единую точку входа. Ее используют для приема внешних запросов и маршрутизации в нужные сервисы внутренней инфраструктуры, отдачи статического контента, аутентификации, стресс тестирования, канареечного развертывания, миграции сервисов, динамического управления трафиком.

Здесь мы будем использовать open-source решение Netflix — Zuul. Spring Cloud нативно интегрирован с ним. В этом проекте Zuul используется для самых элементарных задач — отдачи статики (веб-приложение) и роутинга запросов.

Пример префиксной маршрутизации для Notification service:

```
zuul:
  routes:
    notification-service:
      path: /notifications/**
      serviceId: notification-service
      stripPrefix: false
```

Теперь каждый запрос, uri которого начинается на /notifications, будет направлен в соответствующий сервис.

Service discovery


Еще один широко известный паттерн для распределенных систем. Service discovery позволяет автоматически определять сетевые адреса для доступных экземпляров приложений, которые могут динамически изменяться по причинам масштабирования, падений и обновлений.

Ключевым звеном здесь является Registry service. В этом проекте я использую Netflix Eureka (но есть еще Consul, Zookeeper, Etcd и другие). Eureka — пример client-side discovery паттерна, что означает клиент должен запросить адреса доступных экземпляров и осуществлять балансировку между ними самостоятельно.

Зададим следующую конфигурацию для приложения:

```
spring:
  application:
    name: notification-service
```

Теперь инстанс приложения при старте будет регистрироваться в Eureka, предоставляя мета-данные (такие как хост, порт и прочее). Eureka будет принимать хартбит-сообщения, и если их нет в течении сконфигурированного времени — инстанс будет удален из реестра. Кроме того, Eureka предоставляет дашборд, на котором видны зарегистрированные приложения с количеством инстансов и другая техническая информация: <http://localhost:8761>


HOME
LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2016-04-05T08:25:27 +0000
Data center	default	Uptime	00:03
		Lease expiration enabled	false
		Renews threshold	10
		Renews (last min)	6

DS Replicas

registry

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
ACCOUNT-SERVICE	n/a (1)	(1)	UP (1) - 294720aab0a9:account-service:6000
AUTH-SERVICE	n/a (1)	(1)	UP (1) - 8feb1f6a04a2:auth-service:5000
GATEWAY	n/a (1)	(1)	UP (1) - 4dbf6ede836c:gateway:4000
NOTIFICATION-SERVICE	n/a (1)	(1)	UP (1) - 4e91cd36b09b:notification-service:8000
STATISTICS-SERVICE	n/a (1)	(1)	UP (1) - 070cfc12349a:statistics-service:7000

General Info

Name	Value
total-avail-memory	735mb
environment	test
num-of-cpus	4
current-memory-usage	286mb (38%)
server-uptime	00:03

Клиентский балансировщик, Предохранитель и Http-клиент

Следующий набор инструментов тоже разработан в Netflix и нативно интегрирован в Spring Cloud. Все они работают совместно и используются в микросервисах, которым нужно общаться с внешним миром или внутренней инфраструктурой.

Ribbon

Ribbon — это client-side балансировщик. По сравнению с традиционным, здесь запросы проходят напрямую по нужному адресу, что исключает лишний узел при вызове. Из коробки он интегрирован с механизмом Service Discovery, который предоставляет динамический список доступных инстансов для балансировки между ними.

Hystrix

Hystrix — это имплементация паттерна Circuit Breaker — предохранителя, который дает контроль над задержками и ошибками при вызовах по сети. Основная идея состоит в том, чтобы остановить каскадный отказ в распределенной системе, состоящей из большого числа компонентов. Это позволяет отдавать ошибку как можно быстрее, не задерживаясь при запросе к зависшему сервису (давая ему восстановиться).

Помимо контроля за размыканием цепи, Hystrix позволяет определить fallback-метод, который будет вызван при неуспешном вызове. Тем самым можно отдавать дефолтный ответ, сообщение об ошибке, и др.

На каждый запрос Hystrix генерирует набор метрик (таких как скорость выполнения, результат), что позволяет анализировать общее состояние системы. Ниже будет рассмотрен мониторинг на основе данных метрик.

Feign

Feign — простой и гибкий http-клиент, который нативно интегрирован с Ribbon и Hystrix. Проще говоря, имея в класспase зависимость spring-cloud-

starter-feign и активировав клиент аннотацией `@EnableFeignClients`, мы получаем полный набор из балансировщика, предохранителя и клиента, готовый к работе с некоторой дополнительной конфигурацией.

Пример из Account Service:

```
@FeignClient(name = "statistics-service")
public interface StatisticsServiceClient {

    @RequestMapping(method = RequestMethod.PUT, value = "/statistics/{accountName}", consumes = MediaType.APPLICATION_JSON_UTF8_VALUE)
    void updateStatistics(@PathVariable("accountName") String accountName, Account account);

}
```

Панель мониторинга

Метрики, которые генерирует Hystrix, можно отдавать наружу, включив в класспак зависимость `spring-boot-starter-actuator`. Помимо прочих интересных вещей, будет выставлен специальный эндпоинт — `/hystrix.stream`. Этот стрим можно визуализировать с помощью Hystrix dashboard, о котором мы подробно поговорим ниже. Для включения Hystrix Dashboard понадобится зависимость `spring-cloud-starter-hystrix-dashboard` и аннотация `@EnableHystrixDashboard`. Hystrix Dashboard можно натравить на стрим любого микросервиса для наблюдения живой картины происходящего в этом конкретном сервисе.

Однако в нашем случае сервисов несколько, и необходимо видеть все их метрики в одном месте. Для этого существует специальное решение. Каждый из наших сервисов будет пушить свои стримы в AMQP брокер (RabbitMQ), откуда агрегатор стримов, Turbine, будет преобразовывать их, выставляя единый эндпоинт для Hystrix Dashboard.

Рассмотрим поведение системы под нагрузкой: Account service вызывает Statistics Service, и тот отвечает с варьируемой имитационной задержкой. Пороговое значение времени запроса установлено в 1 секунду.

Success | Short-Circuited | Timeout | Rejected | Failure | Error %

<div><p>Circuit</p><p>account-service.updateStatistics 220 0 0.0 % Host: 21.8/s Cluster: 21.8/s Circuit Closed</p><p>Hosts 1 90th 138ms Median 47ms 99th 316ms Mean 68ms 99.5th 403ms</p><p>Thread Pools</p><p>statistics-service Host: 21.4/s Cluster: 21.4/s</p><p>Active 1 Max Active 9 Queued 0 Executions 214 Pool Size 10 Queue Size 5</p></div>	<div><p>Circuit</p><p>account-service.updateStatistics 105 0 41.0 % Host: 18.7/s Cluster: 18.7/s Circuit Closed</p><p>Hosts 1 90th 569ms Median 545ms 99th 793ms Mean 559ms 99.5th 831ms</p><p>Thread Pools</p><p>statistics-service Host: 10.0/s Cluster: 10.0/s</p><p>Active 5 Max Active 10 Queued 0 Executions 100 Pool Size 10 Queue Size 5</p></div>	<div><p>Circuit</p><p>account-service.updateStatistics 7 0 14.0 % Host: 0.7/s Cluster: 0.7/s Circuit Closed</p><p>Hosts 1 90th 867ms Median 847ms 99th 966ms Mean 855ms 99.5th 975ms</p><p>Thread Pools</p><p>statistics-service Host: 9.7/s Cluster: 9.7/s</p><p>Active 10 Max Active 10 Queued 0 Executions 97 Pool Size 10 Queue Size 5</p></div>	<div><p>Circuit</p><p>account-service.updateStatistics 297 2 100.0 % Host: 20.9/s Cluster: 20.9/s Circuit Open</p><p>Hosts 1 90th 0ms Median 0ms 99th 0ms Mean 0ms 99.5th 0ms</p><p>Thread Pools</p><p>statistics-service Host: 0.2/s Cluster: 0.2/s</p><p>Active 0 Max Active 1 Queued 0 Executions 2 Pool Size 10 Queue Size 5</p></div>
задержка 0 мс	задержка 500 мс	задержка 800 мс	задержка 1100 мс
Система работает без ошибок. Пропускная способность порядка 22 з/с. Небольшое число активных потоков в Statistics service. Среднее время получения ответа — 50 мс.	Число активных потоков увеличивается. Фиолетовая цифра показывает число отклоненных запросов, соответственно порядка 30-40% ошибок, но цепь все еще замкнута.	Полуоткрытое состояние: процент ошибок более 50%, предохранитель размыкает цепь. После определенного таймаута, цепь замыкается, но снова ненадолго.	100% запросов с ошибками. Цепь разомкнута постоянно, попытки пропустить запрос спустя таймаут ничего не меняют — каждый отдельный запрос слишком медленный.

2.2. Функции программного комплекса

Список функций проектируемого программно-аппаратного комплекса:

- Ф1 - предоставление пользовательского интерфейса для взаимодействия с ERP-системой и визуализации бизнес-процессов;
- Ф2 - обеспечение взаимосвязи компонентов исследования: web-интерфейса, серверной части и базы данных;
- Ф3 – интерактивное взаимодействие пользователя с модулями ERP-системы, предоставление CRUD операций для всех объектов системы;
- Ф4 - хранение настроек моделирования и параметров модели в базе данных и конфигурационных файлах;
- Ф5 - мониторинг и визуализация текущего состояния системы и выполняемых операций.

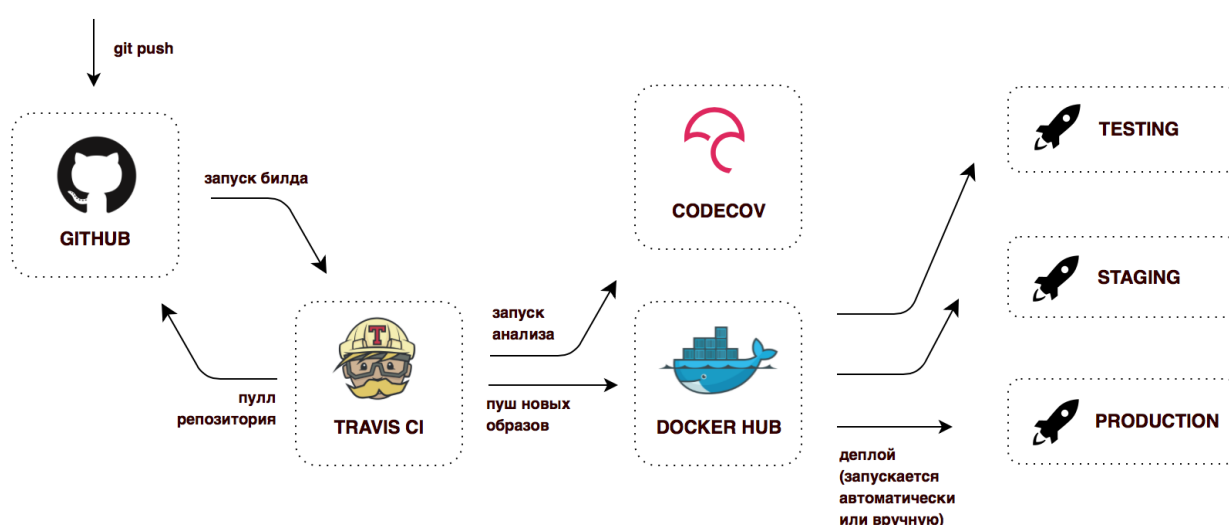
Функция	Подсистемы				
	Frontend Client App	Monitor Dashboard	Account Service	Statistic Service	Zull Gateway
Ф1	+				
Ф2	+				+
Ф3	+		+	+	
Ф4			+		
Ф5	+	+		+	

Таблица 2.5.1 – Матрица инцидентий функций комплекса и функционального назначения подсистем

2.3. Автоматизация инфраструктуры программного комплекса

Развертывание микросервисной системы с большим количеством движущихся частей и взаимосвязанностью — задача очевидно более комплексная, нежели деплой монолитного приложения. Без автоматизированной инфраструктуры вся работа вызовет множество сложностей и большой объем временных затрат.

Схема автоматизации инфраструктуры (CI/CD):



В корне репозитория находится `.travis.yml` файл с указаниями для CI сервера — что делать после удачной сборки. В данной конфигурации на каждый успешный пуш в Github, Travis CI соберет докер-образы, пометит их тегом и запустит в Docker Hub. Теперь получается, что у нас всегда есть готовые к деплою контейнеры, помеченные тегом `latest`, а также контейнеры со старыми версиями, версиями из любых веток.

Production mode

В этом режиме все предварительно собранные образы загружаются из центрального репозитория (в данном случае Docker Hub), порты проброшены наружу докера только для API Gateway, Service Discovery, Monitoring и RabbitMQ management. Все что вам понадобится — это `docker-compose` файл и команда `docker-compose up -d`.

Конфигурационный файл docker-скрипта приведен в Приложении А.

Development mode

В режиме разработки предполагается строить образы, а не забирать их из репозитория. Все контейнеры выставлены наружу для удобного дебага. Эта конфигурация наследуется от приведенной выше, перезаписывая и расширяя указанные моменты. Запускается командой `docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d`

Конфигурационный файл docker-скрипта приведен в Приложении Б.

3 ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

3.1. Основные понятия

Веб-приложение — клиент-серверное приложение, в котором клиентом выступает браузер, а сервером — веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому вебприложения являются кроссплатформенными сервисами [1].

Бэкэнд и фронтэнд термины в программной инженерии, которые различают согласно принципу разделения ответственности между внешним представлением и внутренней реализацией соответственно [2].

3.2. Применение веб-приложений

Веб-приложения позволяют посетителям быстро и легко находить требуемую информацию на веб-сайтах с большим объемом информации. Данный вид веб-приложений позволяет осуществлять поиск в содержимом, упорядочивать содержимое и перемещаться по нему удобным для посетителей способом.

Веб-приложение позволяет сохранять данные непосредственно в базе данных, а также получать данные и формировать отчеты на основе полученных данных для анализа. В качестве примера можно привести интерактивные страницы банков, страницы для контроля товарных запасов, социологические исследования и опросы, а также формы для обратной связи с пользователями [3].

3.3. Структура веб-приложения

Чаще всего веб-приложения состоят как минимум из трёх основных компонентов:

1. Клиентская часть веб приложения - это графический интерфейс. Это то, что вы видите на странице. Графический интерфейс отображается в браузере. Пользователь взаимодействует с веб-приложением именно через браузер, кликая по ссылкам и кнопкам.

2. Серверная часть веб-приложения - это программа или скрипт на сервере, обрабатывающая запросы пользователя (точнее, запросы браузера). Чаще всего серверная часть веб-приложения программируется на PHP. При каждом переходе пользователя по ссылке браузер отправляет запрос к серверу. Сервер обрабатывает этот запрос, вызывая некоторый PHP-скрипт, который формирует веб-страничку, описанную языком HTML, и отправляет клиенту по сети. Браузер тут же отображает полученный результат в виде очередной веб-страницы.

3. База данных (БД, или система управления базами данных, СУБД) - программное обеспечение на сервере, занимающееся хранением данных и их выдачей в нужный момент. В случае форума или блога, хранимые в БД данные - это посты, комментарии, новости, и так далее. База данных располагается на сервере. Серверная часть веб-приложения (то есть, PHP скрипт) обращается к базе данных, извлекая данные, которые необходимы для формирования страницы, запрошенной пользователем.

Браузер через Интернет отправляет HTTP-запросы веб-серверу. Вебсервер вызывает PHP-скрипт, написанный разработчиком веб-приложения. PHP-скрипт обращается к базе данных, если это нужно. В результате PHP-скрипт возвращает клиенту веб-страницу, которую и отображает браузер [4].

3.4. Популярные языки программирования для разработки

Основной язык, которым описывается графический интерфейс вебприложения - это HTML. Данный язык описывает структуру веб-страницы, размещение на ней компонентов.

Оформление веб-страниц, их стиль и цветовая схема описываются в таблицах стилей – CSS. Для "оживления" графического интерфейса, придания ему динамичности, используются дополнительные технологии: скрипты JavaScript, а также встроенные в веб-страницу компоненты, созданные на Flash, Java или Silverlight.

Отсутствие необходимости полностью перезагружать страницу после каждого получения данных от сервера может существенно ускорить работу веб-приложения. Такая концепция имеет название Asynchronous JavaScript and XML (асинхронный JavaScript и XML, Ajax). При использовании данного подхода динамические запросы к серверу происходят без видимой перезагрузки веб-страницы: пользователь не замечает, когда его браузер запрашивает данные [5].

3.5. Доменные модели

Доменные объекты — это объекты в объектно-ориентированных компьютерных программах, выражающие сущности из модели предметной области, относящейся к программе, и реализующие бизнес-логику программы. Доменные объекты инкапсулируют всю необходимую для программы информацию об объекте предметной области [6].

3.6. Роль БД при создании веб-приложений

Согласно классическому определению, база данных — это упорядоченная совокупность информации, хранящейся в виде множеств, каждое из которых содержит записи унифицированного вида. Системы управления базами данных (СУБД) предоставляют программисту мощнейший инструментальный для создания, обновления и обработки больших объемов информации, имеющей сложную структуру.

В классической теории выделяют три типа, три структуры баз данных: иерархическую, сетевую и реляционную. В настоящее время доминирующее положение занимают реляционные базы данных.

Лидером среди баз данных, применяемых для разработки WEBприложений, на сегодняшний день, безусловно, является MySQL. Главное достоинство MySQL (плавно переходящее в недостаток:) - ее простота. Как следствие - высочайшая скорость выполнения SQL-запросов и необходимость явного программирования основных правил поддержания целостности и непротиворечивости данных на уровне сервера приложений.

Среди других баз данных, применяемых для WEB-разработок, отметим Oracle и PostgreSQL. PostgreSQL - свободно распространяемая СУБД с открытым исходным кодом, ориентированная главным образом на работу в UNIX-подобных системах [7].

3.7. Популярные фреймворки для разработки веб-приложений

Фреймворк (framework) — это программная оболочка, позволяющая упростить и ускорить решение типовых задач, характерных для данного языка программирования. Само слово framework означает «каркас» в переводе с английского. Действительно, фреймворки и призваны быть готовыми каркасами программ, на которые только и остаётся навесить стены и окна [8].

Сам фреймворк предлагает нам уже встроенные классы для: работы с базой данных, создания функциональных форм, валидации, логирования и др. Все эти классы можете легко использовать во всех ваших проектах, при этом их подключение и использование будет максимально простым.

Еще один из плюсов - структурирование архитектуры вашего приложения [9].

3.7.1. Maven

Maven (мавен) — это инструмент для сборки Java проекта: компиляции, создания jar, создания дистрибутива программы, генерации документации. Преимущества Maven:

1. Независимость от OS. Сборка проекта происходит в любой операционной системе. Файл проекта один и тот же.

2. Управление зависимостями. Редко какие проекты пишутся без использования сторонних библиотек (зависимостей). Эти сторонние библиотеки зачастую тоже в свою очередь используют библиотеки разных версий. Мавен позволяет управлять такими сложными зависимостями. Что позволяет разрешать конфликты версий и в случае необходимости легко переходить на новые версии библиотек.

3. Возможна сборка из командной строки. Такое часто необходимо для автоматической сборки проекта на сервере (Continuous Integration).

4. Хорошая интеграция с средами разработки. Основные среды разработки на java легко открывают проекты которые собираются с помощью maven. При этом зачастую проект настраивать не нужно - он сразу готов к дальнейшей разработке. Как следствие - если с проектом работают в разных средах разработки, то maven удобный способ хранения настроек. Настроечный файл среды разработки и для сборки один и тот же - меньше дублирования данных и соответственно ошибок.

5. Декларативное описание проекта [10].

3.7.2. Vaadin

Vaadin — свободно распространяемый фреймворк для создания RIA веб-приложений, разрабатываемый одноимённой финской компанией. В отличие от библиотек на Javascript и специфических плагинов для браузеров, Vaadin предлагает сервер-ориентированную архитектуру, базирующуюся на Java Enterprise Edition. Использование JEE позволяет выполнять основную часть логики приложения на стороне сервера, тогда как технология AJAX, используемая на стороне браузера, позволяет интерактивно взаимодействовать с пользователем, не отставая от аналогичных десктопных приложений. Для отображения элементов пользовательского

интерфейса и взаимодействия с сервером на стороне клиента Vaadin использует Google Web Toolkit [11].

Структурно Vaadin состоит из серверного API, клиентского API, набора компонентов пользовательского интерфейса с обеих сторон, механизма тем для оформления интерфейса и модели данных, позволяющей связывать серверные компоненты непосредственно с данными. Можно применять две основные модели разработки: на стороне сервера и на стороне клиента (браузера).

3.7.2. Spring Framework

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Spring Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Одним из компонентов фреймворка является Spring Security, который представляет из себя инструмент, предоставляющий механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для корпоративных приложений, созданных с помощью Spring Framework. [13].

ЗАКЛЮЧЕНИЕ

В ходе практики были закреплены полученные теоретические знания. Получены навыки создания серверных высоконагруженных систем автоматизации управления предприятий с использованием системы ролей, REST API, а также Spring Security, Spring MVC, Spring Rest, Spring Boot, Maven, Hibernate. UnitTesting: JUnit, Mockito, Jersey, PostgreSQL.

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ERP - Enterprise Resource Planning

ПО – Программное обеспечение

УПП - Управление производственным предприятием

БИБЛИОГРАФИЯ

1. Федеральный государственный образовательный стандарт от 12.01.2016 № 5 по направлению подготовки бакалавров 09.03.01 – «Информатика и вычислительная техника».
2. Положение о практике обучающихся (студентов), осваивающих основные профессиональные образовательные программы высшего образования, принятое решением ученого совета Севастопольского государственного университета 24.01.2019.
3. Рабочая программа по технологической (преддипломной) практике студентов по направлению подготовки 09.03.01 «Информатика и вычислительная техника» в лабораториях кафедры ИТиКС и предприятий соответствующего направлению профиля / Сост. И.А. Балакирева. – Севастополь: каф. ИТиКС СевГУ, 2019. – 43с.
4. Методические указания для выполнения типовой выпускной квалификационной работы бакалавра по тематике анализа эффективности компьютерных систем обработки данных для студентов направления 09.03.01(230100) — «Информатика и вычислительная техника; очной и заочной форм обучения / Сост.: И.А. Балакирева, А.В. Скатков – Севастополь: СевГУ, 2015. – 32 с.
5. Методические указания по организации выполнения, оформлению и защите выпускных квалификационных работ бакалавров направления 09.03.01 — Информатика и вычислительная техника; очной и заочной форм обучения /Сост. Балакирева И.А., Брюховецкий А.А., Скатков А.В. — Севастополь, 2016. — 22с.
6. Электронный фонд правовой и нормативно-технической документации. <http://docs.cntd.ru>

ПРИЛОЖЕНИЕ А – Конфигурационный скрипт для стадии production.

```

version: '2'
services:
  rabbitmq:
    image: rabbitmq:3-management
    restart: always
    ports:
      - 15672:15672
    logging:
      options:
        max-size: "10m"
        max-file: "10"

  config:
    environment:
      CONFIG_SERVICE_PASSWORD: $CONFIG_SERVICE_PASSWORD
    image: sqshq/piggymetrics-config
    restart: always
    logging:
      options:
        max-size: "10m"
        max-file: "10"

  registry:
    environment:
      CONFIG_SERVICE_PASSWORD: $CONFIG_SERVICE_PASSWORD
    image: sqshq/piggymetrics-registry
    restart: always
    ports:
      - 8761:8761
    logging:
      options:
        max-size: "10m"
        max-file: "10"

  gateway:
    environment:
      CONFIG_SERVICE_PASSWORD: $CONFIG_SERVICE_PASSWORD
    image: sqshq/piggymetrics-gateway
    restart: always
    ports:
      - 80:4000
    logging:
      options:
        max-size: "10m"
        max-file: "10"

  auth-service:
    environment:
      CONFIG_SERVICE_PASSWORD: $CONFIG_SERVICE_PASSWORD
      NOTIFICATION_SERVICE_PASSWORD: $NOTIFICATION_SERVICE_PASSWORD
      STATISTICS_SERVICE_PASSWORD: $STATISTICS_SERVICE_PASSWORD
      ACCOUNT_SERVICE_PASSWORD: $ACCOUNT_SERVICE_PASSWORD
      MONGODB_PASSWORD: $MONGODB_PASSWORD
    image: sqshq/piggymetrics-auth-service
    restart: always
    logging:
      options:
        max-size: "10m"
        max-file: "10"

```

```

auth-mongodb:
  environment:
    MONGODB_PASSWORD: $MONGODB_PASSWORD
  image: sqshq/piggymetrics-mongodb
  restart: always
  logging:
    options:
      max-size: "10m"
      max-file: "10"

```

```

account-service:
  environment:
    CONFIG_SERVICE_PASSWORD: $CONFIG_SERVICE_PASSWORD
    ACCOUNT_SERVICE_PASSWORD: $ACCOUNT_SERVICE_PASSWORD
    MONGODB_PASSWORD: $MONGODB_PASSWORD
  image: sqshq/piggymetrics-account-service
  restart: always
  logging:
    options:
      max-size: "10m"
      max-file: "10"

```

```

account-mongodb:
  environment:
    INIT_DUMP: account-service-dump.js
    MONGODB_PASSWORD: $MONGODB_PASSWORD
  image: sqshq/piggymetrics-mongodb
  restart: always
  logging:
    options:
      max-size: "10m"
      max-file: "10"

```

```

statistics-service:
  environment:
    CONFIG_SERVICE_PASSWORD: $CONFIG_SERVICE_PASSWORD
    MONGODB_PASSWORD: $MONGODB_PASSWORD
    STATISTICS_SERVICE_PASSWORD: $STATISTICS_SERVICE_PASSWORD
  image: sqshq/piggymetrics-statistics-service
  restart: always
  logging:
    options:
      max-size: "10m"
      max-file: "10"

```

```

statistics-mongodb:
  environment:
    MONGODB_PASSWORD: $MONGODB_PASSWORD
  image: sqshq/piggymetrics-mongodb
  restart: always
  logging:
    options:
      max-size: "10m"
      max-file: "10"

```

```

notification-service:
  environment:
    CONFIG_SERVICE_PASSWORD: $CONFIG_SERVICE_PASSWORD
    MONGODB_PASSWORD: $MONGODB_PASSWORD
    NOTIFICATION_SERVICE_PASSWORD: $NOTIFICATION_SERVICE_PASSWORD
  image: sqshq/piggymetrics-notification-service
  restart: always

```

```
logging:  
  options:  
    max-size: "10m"  
    max-file: "10"
```

```
notification-mongodb:  
  image: sqshq/piggymetrics-mongodb  
  restart: always  
  environment:  
    MONGODB_PASSWORD: $MONGODB_PASSWORD  
  logging:  
    options:  
      max-size: "10m"  
      max-file: "10"
```

```
monitoring:  
  environment:  
    CONFIG_SERVICE_PASSWORD: $CONFIG_SERVICE_PASSWORD  
  image: sqshq/piggymetrics-monitoring  
  restart: always  
  ports:  
    - 9000:8080  
    - 8989:8989  
  logging:  
    options:  
      max-size: "10m"  
      max-file: "10"
```

ПРИЛОЖЕНИЕ Б – Конфигурационный скрипт для стадии разработки.

```
version: '2'
services:
  rabbitmq:
    ports:
      - 5672:5672

  config:
    build: config
    ports:
      - 8888:8888

  registry:
    build: registry

  gateway:
    build: gateway

  auth-service:
    build: auth-service
    ports:
      - 5000:5000

  auth-mongodb:
    build: mongodb
    ports:
      - 25000:27017

  account-service:
    build: account-service
    ports:
      - 6000:6000

  account-mongodb:
    build: mongodb
    ports:
      - 26000:27017

  statistics-service:
    build: statistics-service
    ports:
      - 7000:7000

  statistics-mongodb:
    build: mongodb
    ports:
      - 27000:27017

  notification-service:
    build: notification-service
    ports:
      - 8000:8000

  notification-mongodb:
    build: mongodb
    ports:
      - 28000:27017

  monitoring:
    build: monitorin
```

ПРИЛОЖЕНИЕ В – Листинг программы

```

package com.aleksgolovnya.deansoffice.controller;

import com.aleksgolovnya.deansoffice.entity.Faculty;
import com.aleksgolovnya.deansoffice.repository.FacultyRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.net.URI;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("faculties")
public class FacultyController {

    @Autowired
    private FacultyRepository facultyRepository;

    @GetMapping
    public List<Faculty> retrieveAllFaculties() {
        return facultyRepository.findAll();
    }

    @GetMapping("/{id}")
    public Faculty retrieveFaculty(@PathVariable Long id) {
        Optional<Faculty> faculty =
            facultyRepository.findById(id);

        return faculty.get();
    }

    @DeleteMapping("/{id}")
    public void deleteFaculty(@PathVariable Long id) {
        facultyRepository.deleteById(id);
    }

    @PostMapping
    public ResponseEntity<Object>
        createFaculty(@RequestBody Faculty faculty) {
        Faculty savedFaculty =
            facultyRepository.save(faculty);

        URI location =
            ServletUriComponentsBuilder.fromCurrentRequest().
            path("/{id}")

            .buildAndExpand(savedFaculty.getId()).toUri();

        return ResponseEntity.created(location).build();
    }

    @PutMapping("{id}")
    public ResponseEntity<Object>
        updateFaculty(@RequestBody Faculty faculty,
            @PathVariable Long id) {
        Optional<Faculty> facultyOptional =
            facultyRepository.findById(id);

        if (!facultyOptional.isPresent())
            return ResponseEntity.notFound().build();
    }

```

```

        faculty.setId(id);

        facultyRepository.save(faculty);

        return ResponseEntity.noContent().build();
    }
}

package com.aleksgolovnya.deansoffice.controller;

import
    com.aleksgolovnya.deansoffice.dto.DepartmentDto;
import
    com.aleksgolovnya.deansoffice.entity.Department;
import
    com.aleksgolovnya.deansoffice.repository.DepartmentRepository;
import
    com.aleksgolovnya.deansoffice.service.university.DepartmentService;
import
    org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/departments")
public class DepartmentController {

    @Autowired
    private DepartmentRepository departmentRepository;

    @Autowired
    private DepartmentService departmentService;

    @GetMapping

```

```

    public List<Department> retrieveAllDepartments() {
        return departmentRepository.findAll();
    }

    @GetMapping("/{id}")
    public Department retrieveDepartment(@PathVariable Long id) {
        Optional<Department> department =
            departmentRepository.findById(id);

        return department.get();
    }

    @DeleteMapping("/{id}")
    public void deleteDepartment(@PathVariable Long id) {
        departmentRepository.deleteById(id);
    }

    @PostMapping
    public Department createDepartment(@RequestBody DepartmentDto departmentDto) {
        return
            departmentService.addDepartment(departmentDto);
    }

    @PutMapping("/{id}")
    public Department updateDepartment(@RequestBody DepartmentDto departmentDto, @PathVariable Long id) {
        departmentDto.setId(id);

        return
            departmentService.editDepartment(departmentDto);
    }
}

package com.aleksgolovnya.deansoffice.controller;

```

```

import com.aleksgolovnya.deansoffice.dto.JournalDto;

import com.aleksgolovnya.deansoffice.entity.Journal;

import
    com.aleksgolovnya.deansoffice.repository.JournalRe
        pository;

import
    com.aleksgolovnya.deansoffice.service.subjects.Jour
        nalService;

import
    org.springframework.beans.factory.annotation.Autow
        ired;

import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/journal")
public class JournalController {

    @Autowired
    private JournalService journalService;

    @Autowired
    private JournalRepository journalRepository;

    /** Получить все записи журнала */
    @GetMapping
    public List<Journal> retrieveAllJournals() {
        return journalService.getAll();
    }

    /** Получить конкретную запись журнала */
    @GetMapping("/{id}")
    public Journal retrieveJournal(@PathVariable Long
        id) {
        Optional<Journal> journal =
            journalRepository.findById(id);

```

```

        return journal.get();
    }

    /** Удалить запись журнала */
    @DeleteMapping("/{id}")
    public void deleteJournal(@PathVariable Long id) {
        journalRepository.deleteById(id);
    }

    /** Добавить запись журнала */
    @PostMapping
    public Journal addJournal(@RequestBody
        JournalDto journalDto) {
        return journalService.addJournal(journalDto);
    }

    /** Обновить запись журнала */
    @PutMapping("/{id}")
    public Journal updateJournal(@RequestBody
        JournalDto journalDto, @PathVariable Long id) {
        journalDto.setId(id);
        return journalService.editJournal(journalDto);
    }
}

package com.aleksgolovnya.deansoffice.controller;

import
    com.aleksgolovnya.deansoffice.dto.ScheduleDto;

import
    com.aleksgolovnya.deansoffice.entity.Schedule;

import com.aleksgolovnya.deansoffice.entity.Student;

import
    com.aleksgolovnya.deansoffice.repository.ScheduleR
        epository;

import
    com.aleksgolovnya.deansoffice.repository.StudentRe
        pository;

```



```

import
    com.aleksgolovnya.deansoffice.service.schedule.Sch
    eduleService;

import
    com.aleksgolovnya.deansoffice.service.schedule.Sch
    eduleServiceImpl;

import org.modelmapper.ModelMapper;

import
    org.springframework.beans.factory.annotation.Autow
    ired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.*;

import
    org.springframework.web.servlet.support.ServletUri
    ComponentsBuilder;

import java.net.URI;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/schedule")

public class ScheduleController {

    @Autowired

    private ScheduleService scheduleService;

    @Autowired

    private ScheduleRepository scheduleRepository;

    @GetMapping

    public List<Schedule> retrieveAllSchedules() {

        return scheduleService.getAll();

    }

    @GetMapping("/{id}")

    public Schedule retrieveSchedule(@PathVariable
    Long id) {

```

```

        Optional<Schedule> schedule =
        scheduleRepository.findById(id);

        return schedule.get();

    }

    @GetMapping("/lessons/{id}")

    public List<Schedule>
    getTeacherLessons(@PathVariable Long id) {

        List<Schedule> lessons =
        scheduleService.getTeacherLessons(id);

        return lessons;

    }

    @DeleteMapping("/{id}")

    public void deleteSchedule(@PathVariable Long id)
    {

        scheduleRepository.deleteById(id);

    }

    @PostMapping

    public Schedule addSchedule(@RequestBody
    ScheduleDto scheduleDto) {

        return
        scheduleService.addSchedule(scheduleDto);

    }

    @PutMapping("/{id}")

    public Schedule updateSchedule(@RequestBody
    ScheduleDto scheduleDto, @PathVariable Long id) {

        scheduleDto.setId(id);

        return
        scheduleService.editSchedule(scheduleDto);

    }

}

package com.aleksgolovnya.deansoffice.controller;

```

```

import
    com.aleksgolovnya.deansoffice.dto.SpecialtyDto;

import
    com.aleksgolovnya.deansoffice.entity.Specialty;

import
    com.aleksgolovnya.deansoffice.repository.SpecialtyRepository;

import
    com.aleksgolovnya.deansoffice.service.university.SpecialtyService;

import
    org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import
    org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import java.net.URI;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/specialties")
public class SpecialtyController {

    @Autowired
    private SpecialtyRepository specialtyRepository;

    @Autowired
    private SpecialtyService specialtyService;

    @GetMapping
    public List<Specialty> retrieveAllSpecialties() {
        return specialtyRepository.findAll();
    }

    @GetMapping("/{id}")

```

```

    public Specialty retrieveSpecialty(@PathVariable Long id) {

        Optional<Specialty> specialty =
            specialtyRepository.findById(id);

        return specialty.get();
    }

    @DeleteMapping("/{id}")
    public void deleteSpecialty(@PathVariable Long id) {
        specialtyRepository.deleteById(id);
    }

    @PostMapping
    public Specialty createSpecialty(@RequestBody SpecialtyDto specialtyDto) {
        return
            specialtyService.addSpecialty(specialtyDto);
    }

    @PutMapping("/{id}")
    public Specialty updateSpecialty(@RequestBody SpecialtyDto specialtyDto, @PathVariable Long id) {
        specialtyDto.setId(id);

        return
            specialtyService.editSpecialty(specialtyDto);
    }
}

package com.aleksgolovnya.deansoffice.controller;

import
    com.aleksgolovnya.deansoffice.dto.SpecialtyDto

```