МИНИСТЕРСТВО НАУКИ И ВЫСШЕГООБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БАЛТИЙСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. И. КАНТА»
ИНСТИТУТ ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Рекомендована к защите:
методический руководитель направления подготовки
доцент ИФМНиИТ

_____ Д.А. Савкин

"____"_____ 2020 г.

Допущена к защите:
первый заместитель директора ИФМНиИТ,
к. ф.-м. н., доцент

_____А.А. Шпилевой

"____"_____2020 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Тема: «Особенности тестирования производительности информационных систем планирования и анализа бюджета муниципального образования»

Направление подготовки: 01.04.02. «Прикладная математика и информатика»

Магистерская программа: «Информационные системы в государственном и муниципальном управлении»

Квалификация (степень**)**: **магистр**

ВКР защищена на оценку: _____

Выполнил: студент 2 курса
_____ И.В. Савчук

Руководитель: к.ф.-м.н., доцент ИФМНиИТ
_____ Б.Р. Мищук

Рецензент: Руководитель направления автоматизированного и нагрузочного тестирования ООО «БФТ»

_____ Е.А. Разиньков

Калининград, 2020

Оглавление

Введение

Рынок российского программного обеспечения (ПО) быстро развивается, и не только в потребительском сегменте, но и в сфере госзакупок. Так, на 24 апреля 2019 года стало известно, что доля отечественного ПО в закупках госорганов составляет на апрель 2019 года 65%, в то время как в 2015 г. она достигала всего 20%. Такие данные привел заместитель главы Минкомсвязи Алексей Соколов[1].

Такой скачок стал следствием постановление Правительства РФ от 16 ноября 2015 г. № 1236 “Об установлении запрета на допуск программного обеспечения, происходящего из иностранных государств, для целей осуществления закупок для обеспечения государственных и муниципальных нужд” [2] и подписанием Президентом России Федерального закона от 29 июня 2015 г. N 188-ФЗ "О внесении изменений в Федеральный закон "Об информации, информационных технологиях и о защите информации" и статью 14 Федерального закона "О контрактной системе в сфере закупок товаров, работ, услуг для обеспечения государственных и муниципальных нужд"[1]. Вследствие чего востребованность финансовых программных продуктов компании ООО "Бюджетные и финансовые технологии" возрастает с каждым годом, так же как возрастает и нагрузка на эти системы, когда с ними работает все больше людей.

Любые ошибки в работе сложной финансовой системы в первую очередь могут привести к значительным финансовым проблемам и утрате доверия к поставщику ПО. По это

причине необходимо тщательное тестирование всей системы перед выдачей ее заказчику[16]. И наряду с функциональным тестированием не менее важно тестирование производительности, так как с системой впоследствии будут работать сотни и тысячи людей одновременно, создавать и обрабатывать электронные документы. И здесь важна как скорость работы, так и стабильность работы всей системы и ее отдельных компонентов.

По расчетам Минкомсвязи, к 2024 г. доля российского ПО будет доведена до 90% в закупках госорганов, и до 70% — в закупках госкомпаний. На 2019 г. для госкомпаний запланирован показатель на уровне не менее 45%.

Министерство также привело актуальные данные о Реестре отечественного ПО, который ведет с 2016 г. — на апрель 2019 года там насчитывается более 5,2 тыс. программных продуктов в 24 различных классах.

Целью данной выпускной квалификационной работы является разработка схемы комплексного тестирования производительности информационной системы, предназначенной для планирования и анализа бюджета с целью улучшения качества программного комплекса «АЦК-Планирование».

Для достижения данной цели прославлены следующие задачи:

- Изучить особенности электронного бюджетирования РФ;

- Исследовать инструменты для тестирования производительности и анализа результатов тестирования;
- Разработать схему и провести комплексное тестирование производительности системы «АЦК-Планирование»;
- Проанализировать результаты тестирования производительности.

Практическая значимость выпускной квалификационной работы определяется тем, что с разработанной схемой уменьшаются время и затраты на тестирование производительности, а также повышается качество и стабильность выпускаемого ПО.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка использованных источников и приложения.

1. Бюджетная система РФ

1.1. Понятие бюджетного механизма

В отечественной экономической литературе вопросы формирования и функционирования бюджетного механизма недостаточно исследованы и обоснованы. Понятие «бюджетный механизм» часто отождествляют с финансовым механизмом, не совсем правильно, поскольку, являясь составной последнего, бюджетному механизму присущи специфические черты, связанные с особенностями функционирования бюджетных отношений, специфическими формами и методами мобилизации и использования бюджетных средств.

Конечно, бюджетный механизм имеет отдельные черты финансового механизма. Так, бюджетный механизм, характеризуя сферу использования объективно существующих отношений, выступает составным звеном общей системы управления экономикой, занимая в ней свое место. Также бюджетный механизм, с одной стороны, обусловлен реальным существованием бюджетных отношений, с другой – обратно активно влияет на них.

Наряду с общими чертами финансового и бюджетного механизма, последний имеет в собственности и свои особенности, выражающиеся в совокупности определенных видов бюджетных отношений, специфических методов мобилизации и использования бюджетных средств.

Подходы к толкованию категории «бюджетный механизм» отмечаются разным уровнем детализации его

составляющих. В широком понимании бюджетный механизм – это «совокупность форм, методов, рычагов и инструментов использования государственного бюджета и влияния на социально-экономическое развитие".

Сущность бюджетного механизма можно определить двумя подходами к этому понятию. С одной стороны, бюджетный механизм можно понимать как функционирование самых бюджетных средств. Материальным отражением финансовых отношений являются денежные потоки. Организация этих потоков, порядок их осуществления происходят за определенными правилами и направлениями. В этом случае бюджетный механизм отражает внутреннюю организацию функционирования финансовых ресурсов. С другой стороны, бюджетный механизм рассматривается как совокупность методов и форм, инструментов, приемов и рычагов влияния на состояние и развитие экономики. Этот подход отражает внешнее воздействие функционирования бюджета и характеризует финансовые ресурсы как фактор влияния на состояние экономики [9].

Д. В. Дементьев трактует бюджетный механизм как "совокупность форм и методов, рычагов и инструментов мобилизации и использования бюджетных средств" [10]. Курченко Л.Ф. рассматривает бюджетный механизм как категорию управления: "Бюджетный механизм можно назвать механизмом управления бюджетным процессом, реализации на практике функций и принципов бюджетной системы и бюджетной политики на базе норм бюджетно-налогового права" [11].

Подробнее бюджетный механизм трактуют как "совокупность конкретных форм бюджетных отношений, специфических методов мобилизации и использования бюджетных средств государством. Он является реальным воплощением бюджетной политики и отражает конкретное направление бюджетных отношений на выполнение экономических и социальных задач каждого исторического этапа развития страны".

Итак, по моему мнению, бюджетный механизм выступает как сложная и динамичная экономическая категория, которая, с одной стороны, выступает как совокупность форм, методов и мероприятий организации бюджетных отношений в государстве, а с другой стороны, материальным воплощением финансовых отношений и их практического применения для достижения соответствующих целей и задач, определенных бюджетной политикой государства.

Бюджетный механизм – это практическое использование бюджета для осуществления финансовой политики государства. С помощью бюджетного механизма можно практически использовать бюджет как инструмент государственного регулирования экономики, стимулирования производственных и социальных процессов, то есть следует заметить, что бюджетный механизм выступает как структурированная система и имеет практическую реализацию.

Некоторые ученые считают, что "через бюджетный механизм государство регулирует экономику, стимулирует производственные и социальные процессы".

1.2. Структура бюджетного механизма РФ

С целью более глубокого раскрытия сути бюджетного механизма целесообразно рассмотреть его функциональную структуру, то есть функции каждого элемента в системе.

Исследуя бюджетный механизм, Ю. Пасечник определяет его составляющие (бюджетное планирование и регулирование, финансовые показатели, нормативы, лимиты, резервы, систему управления бюджетными средствами) и их сущность.

Можно выделить некоторые составляющие, которые действуют как единое целое с характерным комплексом только им присущих черт.

Прежде всего, это блок рычагов по мобилизации бюджетных ресурсов, что находит свое отражение в конкретных видах налогов и платежей.

Следующий блок ориентирует на использование бюджетных ресурсов через принципы и методы бюджетного финансирования.

И, наконец, блок бюджетного регулирования, обеспечивает функционирование всей бюджетной системы.

В каждом из звеньев есть элементы: конкретные виды бюджетных расходов, доходов и поступлений (см. рисунок 1).

Рисунок 1. Структура бюджетного механизма

Функционируют звенья бюджетного механизма с помощью методов (приемов), способов, условий, определяющих объем и движение бюджетных ресурсов.

В структуре бюджетного механизма также могут выделяться отдельные блоки, что обусловлено функциональными особенностями управления бюджетными отношениями:

– бюджетного планирования и бюджетного прогнозирования;

– исполнения бюджета;

– бюджетного контроля и т.д. [12].

Каждый из указанных блоков характеризуется принципами, специфическими методами (способами, приемами), с помощью которых решаются задачи, возникающие на каждой стадии бюджетного процесса:

1) механизм регионального бюджетного планирования. Он предназначен для осуществления бюджетного планирования на уровне региона и включает процессы составления, рассмотрения и утверждения бюджета, а также методологию и методику их осуществления. С помощью особых приемов и методов, используемых в блоке бюджетного планирования, составляется финансовый план, который отражает мобилизацию и расходование бюджетных средств на уровне региона.

В РФ местный бюджет, также как и федеральный, планируется на один финансовый год. Основными принципами планирования бюджета по Бюджетному кодексу являются:

– принцип преемственности;

– принцип приоритетности;

– принцип обоснованности.

Порядок разработки проекта регионального бюджета определяется Правительством РФ. Основой для разработки проектов местных бюджетов на предстоящий финансовый год являются (см. рисунок 2):

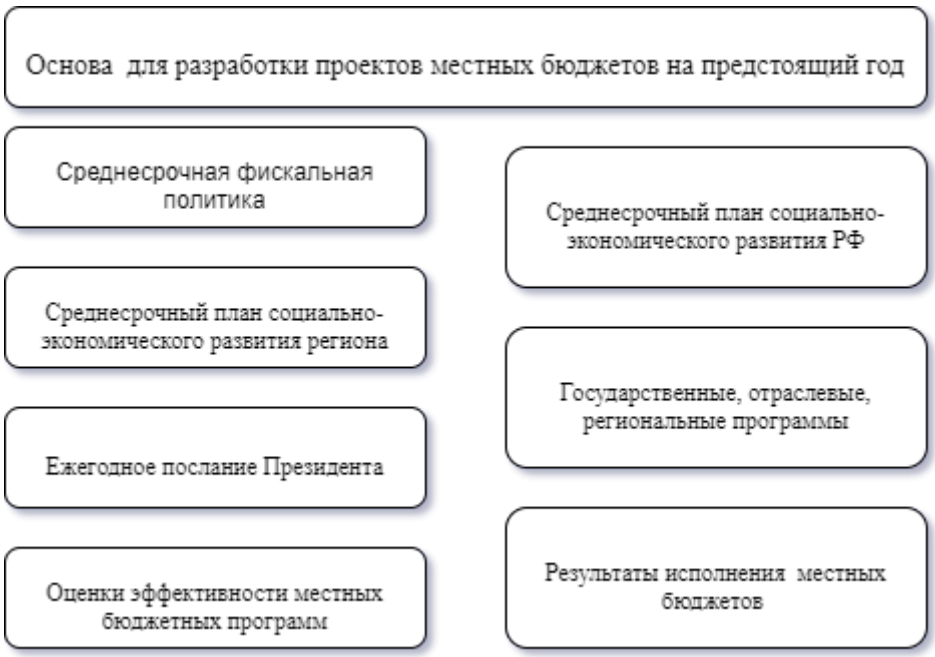

Рисунок 2. Список документов, являющихся основой для разработки проектов местных бюджетов

Одновременно с проектом бюджета на очередной финансовый год составляется прогноз бюджета на предстоящий трехлетний период.

Структура бюджетного механизма отражает следующие основные группы бюджетных отношений, как бюджетные нормы и нормативы, бюджетные стимулы, бюджетные резервы в соответствии с ним существуют различные формы и методы перераспределения денежных средств между сферами их использования. Поэтому в бюджетном механизме, в зависимости от форм проявления бюджетных отношений, следует выделить следующие структурные звенья: методы мобилизации денежных средств в государственный бюджет; формы и порядок предоставления бюджетных средств субъектам хозяйствования; способы межбюджетного распределения и перераспределения

финансовых ресурсов и маневрирования ими. Перечисленные звена бюджетного механизма, в свою очередь, содержат соответствующие элементы. Так, бюджетный механизм мобилизации денежных средств характеризуется налоговыми и неналоговыми методами поступления финансовых ресурсов, получает свое выражение в конкретных видах налогов – налога на добавленную стоимость, акцизного сбора, налога на прибыль и других. Механизм предоставления бюджетных средств предприятиям, организациям и учреждениям реализуется с помощью соответствующего финансирования и обеспечения денежными средствами. Механизм межбюджетного распределения и перераспределения финансовых ресурсов характеризуется закреплением доходов по надлежащим звеньями бюджетной системы и различными методами бюджетного регулирования — дотациями, субвенциями.

Каждому структурному звену и элементу бюджетного механизма соответствуют свои функциональные особенности, они обусловлены теми задачами и целями, для решения которых они предназначены. Несмотря на относительную самостоятельность, элементы и звенья бюджетного механизма функционируют как единое целое. Именно взаимосвязь действия всех структурных звеньев и элементов обеспечивает синхронность функционирования бюджетного механизма в целом [15].

Научные подходы к пониманию бюджетного механизма требуют пересмотра и адаптации к социально-экономическому положению государства в современной среде, вследствие чего объективно возникла потребность

вернуться к анализу понятия и структуры бюджетного механизма.

Анализируя вышеупомянутые взгляды относительно структуры бюджетного механизма, на мой взгляд, целесообразно выделить оптимальные его составляющие, которые в полной мере будут отражать его цели и назначения в общенациональной политике государства.

Методы бюджетного механизма олицетворяют в себе средства воздействия, с помощью которых государство осуществляет организацию бюджетных отношений и наиболее эффективное формулирование размещения и использования бюджетных ресурсов. К методам бюджетного механизма относится бюджетное планирование и прогнозирование, бюджетный контроль, бюджетное обеспечение, оперативное управление бюджетными средствами, финансирование и налогообложение.

Рычаги бюджетного механизма характеризуют, как установлена государством система средств, которая применяется для практической реализации задач и мероприятий, предусмотренных бюджетной политикой. Использование методов бюджетного механизма невозможно без соответствующих рычагов, среди которых целесообразно выделить: налоги и сборы, межбюджетные трансферты, бюджетные стимулы и санкции.

Инструменты бюджетного механизма функционируют в разрезе бюджетных рычагов, которые конкретизируют их функциональное назначение и является самым мобильным составляющим бюджетного механизма. Каждый из бюджетных рычагов характеризуется определенной

совокупностью инструментов, которые в целом формируют инструменты бюджетного механизма и к ним относят ставки налогов, нормативы, лимиты, резервы, штрафы, льготы, ставки заработной платы.

Роль информационной составляющей является основой быстрой реакции на ситуацию и необходимости внесения коррективов в финансовый и бюджетный механизмы на современном этапе развития ей уделяется все больше внимания. Это означает, что наряду с традиционными инструментами-регуляторами бюджетной политики во внимание стали приниматься также новые информационные инструменты, в частности, открытость и доступность информации о состоянии бюджетной системы и социально-экономических процессах в целом для каждого члена общества.

Итак, бюджетный механизм включает взаимодействие бюджетных методов, инструментов и рычагов, с помощью которых определяются оптимальные объемы формирования и использования для обеспечения социально – экономического развития общества [2].

То есть, выступая составным звеном общей финансовой системы государства, бюджетный механизм сводится к выполнению двух основных функций: финансового регулирования экономикой и социальными процессами и финансового их обеспечения. Поэтому функционирования только одного бюджетного механизма не может обеспечить оптимального объема формирования ресурсов бюджетов различных уровней и их рационального использования, эффективность функционирования бюджетного механизма

может быть достигнута путем согласованного взаимодействия всех составляющих хозяйственного механизма. Чрезвычайно важным является обеспечение согласования мероприятий по совершенствованию всех составляющих хозяйственного механизма, направленных на повышение эффективности общественного производства.

1.3. Этапы и особенности распределения бюджета РФ

Бюджетная система РФ предоставляет средства, с помощью которых правительство решает, сколько денег потратить и на что потратить, и как собрать деньги, которые оно решило потратить. После принятия этих решений бюджетная система обеспечивает их выполнение. Правительство использует бюджетную систему для определения распределения ресурсов между его основными функциями, такими как обеспечение национальной обороны, развитие торговли и оказание медицинской помощи, а также для определения целей и масштабов отдельных программ, проектов и видов деятельности. Хотя бюджетная система сосредоточена на рублях, другие ресурсы, такие как федеральная занятость, также контролируются через бюджетную систему. Решения, принятые в бюджетном процессе, затрагивают нацию в целом, государственные и местные органы власти и отдельных россиян. Многие бюджетные решения имеют всемирное значение.

Составление прогноза бюджета состоит из следующих этапов и имеет итеративный характер.

1 этап. Минэкономики РФ вместе с другими федеральными органами исполнительной власти, принимая во внимание учет Послания Президента РФ Федеральному Собранию РФ (которое вышло в этом году) начинает разработку сценарных условий функционирования государственной экономики. В них содержатся основные параметры, целевые решения и действия по стабилизации по областям государственной политики, дается оценка ключевых показателей. Показатели по сценарным условиям экономического функционирования разрабатываются в соответствии с новыми тенденциями по социально-экономическому развитию, анализа прогнозов предыдущих лет. Разбираются разные, но наиболее ожидаемые варианты, оценивают также остроту влияния факторов неблагоприятного характера [1].

В соответствии с полученными результатами выделяют уже **основной вариант (оптимистичный)**, который сориентированный на возможной полной реализации целей и задач. В нем, при этом, не учитываются непредсказуемые изменения ни экономической конъюнктуры в мире, ни политической ситуации внутри страны. В **другом варианте (пессимистичном)** учитывается влияние возможных негативных факторов, которые могут привести к замедлению выхода на намеченные цели в определенные сроки. Для того, чтобы устранить влияние этих факторов нужны дополнительные затраты, поэтому темпы экономического развития будут ниже ожидаемых. Минфин РФ,

Минэкономразвития РФ и ЦБ РФ, синхронно с подготовкой сценарных условий, представляют рекомендации по направлениям бюджетной политики.

2 этап. После того, как Правительством РФ будут одобрены сценарные условия и направления бюджетной политики, Минэкономразвития РФ доводит их до субъектов РФ и органов исполнительной власти РФ. Федеральные ведомства и министерства начинают подготавливать свои предложения, принимая во внимание отраслевые особенности. Так, Федеральной комиссией по рынку ценных бумаг подготавливаются предложения по развитию рынка ценных бумаг; Министерством здравоохранения и социального развития определяются главные направления по развитию социальных процессов, политики доходов, обеспечения занятости и т.п.

Система органов исполнительной власти субъектов РФ представляет федеральным органам, которые выступают в качестве государственных заказчиков федеральных целевых программ, рекомендации по их осуществлению на территории определенных субъектов РФ. Для того, чтобы спрогнозировать консолидированный бюджет, они также представляют по показателям бюджетной классификации отчеты о результатах исполнения бюджетов за прошлый год и ратифицированные бюджеты на следующий год.

Минэкономразвития РФ всю эту информацию обобщает и передает в Правительство РФ. В то же время Минэкономразвития РФ передает в Минфин РФ главные показатели по прогнозу российского социально-экономического развития в целом и по государственным

субъектам, а также список межгосударственных и федеральных целевых программ, которые предлагаются к финансированию из бюджета государства, указывая проект сводной бюджетной заявки, а также источники и объемы финансирования.

3 этап. После того, как Правительство РФ рассмотрит всю информацию, Минэкономразвития РФ передает органам исполнительной власти РФ и субъектам РФ уточненную информацию по предварительному прогнозу, необходимую для разработки региональных и отраслевых прогнозов, а также для того, чтобы спроектировать бюджетные ассигнования по инвестиционным программам и федеральным целевым.

4 этап. Исполнительные органы власти и субъектов РФ, а также государственные заказчики федеральных целевых программ передают уточненные прогнозы по социально-экономическому развитию отраслей экономики и регионов в Минэкономразвития РФ принимая во внимание результаты предварительного рассмотрения, оценку экономического состояния, возможные варианты стабилизации экономики, а также уточняются расходы на осуществление государственных целевых программ отталкиваясь от объемов их финансирования, которые уточнены в федеральном бюджете.

5 этап. Минэкономразвития РФ передает в Правительство конечный прогноз по социально-экономическому развитию РФ, проект по сводному финансовому балансу с разбивкой по субъектам РФ; объемы поставок продукции для федеральных нужд; список

первостепенных социально-экономических проблем (задач), которые должны решиться в следующем году, и государственных целевых программ, которые предлагаются к финансированию за счет государственного бюджета; проект по развитию экономического государственного сектора и пр.

В соответствии с прогнозом Правительством РФ происходит утверждение основных показателей:

- объема и динамики ВВП;
- прогноза инфляции;
- объемов производства и реализации продукции, работ и услуг;
- объемов инвестиций в основной капитал по всем источникам финансирования;
- расчетов фонда заработной платы;
- объемов доходов (прибыли);
- показателей импорта и экспорта;
- прогноза ставок рефинансирования ЦБ РФ и курса рубля;
- проектировок главных характеристик по сводному финансовому балансу.

1.4. Использование электронной бюджетной системы

Использование ИКТ (информационно-коммуникационных технологий) в государственном секторе охватывает все виды государственной деятельности, включая бюджетную сферу. Термин «электронное бюджетирование» используется для обозначения любого приложения или

инструмента ИКТ, которые используются для бюджетных функций, процедур или услуг в течение бюджетного цикла (планирование, составление бюджета, ассигнования, контроль и оценка финансовых ресурсов).

Электронное бюджетирование включает в себя три различных, хотя и связанных, профиля: первый и основной состоит из оцифровки бюджетных процедур; второй связан с распространением бюджетной информации на публике в открытом формате (открытые данные); третий касается использования комплексных баз данных (БД) бюджетной информации для формирования политики (большие данные).

Автоматизированные решения, позволяющие государственным учреждениям планировать, исполнять и контролировать бюджет, обычно называются системами финансового управления и информации (ИСФМ). ИСФМ помогает лицам, принимающим решения, соблюдать финансовые правила и стандарты отчетности. Кроме того, платформы ИСФМ облегчают раскрытие информации, касающейся бюджетной процедуры, и, следовательно, потенциально могут оказать благотворное влияние на подотчетность и участие правительства.

Есть много примеров использования электронных инструментов в бюджетной процедуре. Интересное использование электронных инструментов можно наблюдать в области составления бюджета на основе участия - то есть процесса, в котором члены сообщества принимают непосредственное участие в принятии решений о том, как расходовать часть государственного бюджета. Использование

ИКТ в бюджетировании с участием населения в основном направлено на расширение участия граждан.

Первое преимущество, связанное с оцифровкой бюджетных процедур — это экономия средств. Проектирование государственных услуг (включая бюджет) в цифровом формате может сэкономить от 6,5 до 10 миллиардов евро в год. Например, оцифровка процедур государственных закупок может сократить расходы на государственное управление на 15-20%, при этом экономия составит около 100 млрд. евро в год.

Вторым преимуществом электронного бюджетирования является снижение административного бремени. Это бремя состоит из затрат, которые несут граждане и бизнес, когда они обязаны выполнять бюрократические требования, установленные государственным регулированием. Ожидается, что внедрение цифровых процедур поможет установить процедуры, повысить административную эффективность и сократить расходы для граждан и предприятий. Реализация политики единовременной регистрации означает, что пользователи будут вводить свою личную информацию в административных целях.

Более того, снижение административного бремени влияет на удовлетворенность граждан государственными услугами.

Второе значение электронного бюджетирования заключается в распространении бюджетной информации через Интернет в открытом формате. Это очень чувствительный политический вопрос. То, как государственные деньги собираются и распределяются,

является предметом общего интереса: на всех влияет принятие решений, касающихся фискальной / бюджетной политики. Как следствие, распространение информации о распределении государственных средств играет жизненно важную роль в преодолении дефицита отчетности государственных учреждений.

Цифровые технологии считаются потенциальным решением для повышения прозрачности в бюджетных вопросах. «Открытые данные» — это концепция, используемая для обозначения наборов данных, которые можно свободно использовать и распространять без каких-либо юридических или технологических ограничений. Поскольку доступность Интернета продолжает увеличиваться, ожидается, что спрос на увеличение объемов (и качества) информации, касающейся деятельности органов государственной власти, будет усиливаться. Эти запросы на информацию будут включать данные о бюджетах, которые включают бюджетные данные, а также другие компоненты правительственной финансовой деятельности, такие как дополнительные бюджетные средства, налоговые расходы и условные обязательства. Это также демонстрируется растущим использованием термина «открытые бюджетные данные» и его лексических вариантов в Интернете и социальных сетях [12].

Большинство правительств вложили значительные средства в наращивание потенциала и технологии для разработки инструментов для выпуска информации о бюджетном цикле в открытом формате.

По данным Всемирного банка, в мире существует 48 стран, где гражданское общество и граждане имеют возможность воспользоваться информацией о государственных финансах, публикуемой в Интернете, для мониторинга бюджета и обеспечения подотчетности своих правительств. Однако прогресс неравномерен. Всемирный банк сообщает, что «правительства стран с высоким и средним уровнем дохода динамически публикуют бюджетные данные в различных форматах, в основном из централизованных систем». В Европе примеры Германии и Великобритании являются образцовыми. В Германии система государственного бюджета подлежит обязательной прозрачности и публикации на всех административных уровнях. Например, штат Баден-Вюртемберг публикует и связывает все документы, связанные с бюджетным процессом. С другой стороны, страны с низким уровнем дохода, как правило, публикуют статические бюджетные данные, в основном через документы, размещаемые на веб-сайтах государственных финансов.

Но несмотря на то, что цифровизация активно проникает в государственную экономику, имеет полную государственную и президентскую поддержку, проводится активная деятельность в области мобильной государственной идеи в СМИ, имеет место постоянное обсуждение самых результативных методов и способов организации процесса на разных совещаниях и форумах, цифровизация проходит крайне медленно. Существует большое количество технологических проблем и проблем, которые непосредственно связаны с интеграцией.

Информационная или цифровая экономика имеет большое количество возможностей в области построения бюджета, области общественных финансов и управления государством, благодаря:

- глубокому анализу имеющейся финансовой ситуации в государстве и стратегическое планирование с построением экономических моделей;

- мониторингу, координации и контролю всех взаимоотношений в области общественных финансов;

- принципиально новому уровню предотвращения появления коррупционных элементов на фоне доступности и открытости информации;

- своевременном реагировании и оперативном управлении в возможности возникновения экономических кризисов.

Глава 2. Тестирование производительности

Создание успешного продукта зависит от двух основных компонентов - функциональности и производительности. К «Функциональности» относится набор возможностей, которые приложение предоставляет своим пользователям, включая транзакции, которые оно делает, и информацию, которую оно делает доступной. «Производительность» относится к способности системы совершать транзакции, а также быстро и точно предоставлять информацию, несмотря на высокое многопользовательское взаимодействие или ограниченные аппаратные ресурсы.

Сбой приложения из-за проблем с производительностью можно предотвратить с помощью тестирования производительности перед выпуском программного продукта.

3.1 Концепции производительности

Тестирование производительности — это набор типов тестирования, предназначенных для воссоздания пользовательских запросов к системе и сравнения ожидаемых результатов с полученными показателями, а также для определения скорости процедур, стабильности, надежности и масштабируемости системы в целом. Полученные результаты позволяют находить узкие места в системе. Узкие места — это единственная точка или компонент в системы, которая сдерживает всю

производительность[7]. Например, даже самый быстрый компьютер будет плохо работать в современной сети, если пропускная способность меньше 1 мегабита в секунду. Низкая скорость передачи данных может быть присуща аппаратному обеспечению , но также может возникать из-за проблем, связанных с программным обеспечением. Таким образом, тестирование производительности позволяет выявить уязвимости с пропускной способностью приложения, временем загрузки, обработкой больших объемов данных и предотвращать их в приложении[30].

Тестирование производительности сильно помогает в разработке ПО, например[5]:

- Повышает надежность: тестирование производительности помогает избежать взаимных блокировок, улучшить время отклика; проверяет масштабируемость, отказоустойчивость, восстановление после сбоев и т. д.
- Сокращает времени выхода продукта на рынок: тестирование производительности значительно сокращает время выхода на рынок крупных корпоративных приложений. В целом, если 98% критических блоков были успешно протестированы, то считается, что пора выпустить на рынок.
- Локализовать «узкие места» системы. Такие как утечки памяти; ошибки, связанные с накоплением данных; ограничения пропускной способности сетевого канала; нерациональное использование аппаратных ресурсов и др.

- Тестирование производительности помогает повысить безопасность ПО путем обнаружения переполнения памяти и других уязвимостей ресурсов как для веб-приложений, так и для настольных;
- Произвести сравнительный анализ. К примеру, проверить работу системы на разных операционных системах с различными конфигурациями аппаратных характеристик и т.д.;
- Позволяет проверить масштабируемость приложения, чтобы точно предсказать требуемые аппаратные характеристики системы и сети, что, в свою очередь, поможет избежать проблем в будущем при расширении инфраструктуры.

Ресурсные аспекты, такие как использование ЦП, оперативной памяти, когерентность кэша, согласованность данных, пропускная способность сети, также отслеживаются и регистрируются как часть тестирования производительности. Кроме того, время отклика веб-сервера, сервера приложения и сервера БД также учитываются. В совокупности производительность системы воспринимается как показатель качества с точки зрения времени отклика, пропускной способности, доступности, надежности, безопасности, масштабируемости и расширяемости.

3.2 Виды тестирования производительности

Существует множество видов тестирования производительности. Классификация видов тестирования производительности строится на основе того, какие цели преследует определенный вид тестирования[17]. Как правило тестирование производительности преследует не одну, а несколько целей.

Это связано с тем, что многие типы тестирования в ходе его проведения совмещаются с другими целями или повторяются несколько раз на протяжении всего цикла тестирования. Однако тестирование производительности, в отличие от остальных типов тестирования, проводится только после полного функционального тестирования[33]. В ходе тестирования производительности ошибки функциональности не исправляются. Для данного вида тестирования, как правило, выделяется отдельный нагрузочный стенд, являющийся копией промышленного стенда.

Таким образом, основная классификация видов тестирования производительности представлена на рисунке 3:

Рисунок 3. Классификация видов тестирования производительности

На основе классификации из Рисунка 4 можно выделить основные виды тестирования производительности, которые проводятся чаще всего. Основные типы тестирования, а также вопросы, которые они решают можно представить в виде таблицы 1[14]:

Таблица 1. Вида тестирования производительности

Вид тестирования	Вопрос, на который отвечает тестирование
Нагрузочное тестирование (Load Testing)	Достаточно ли быстро работает система?
Тестирование стабильности (Stability Testing)	Достаточно ли надежно работает система на долгом интервале времени?
Стрессовое тестирование (Stress Testing)	Что произойдет при незапланированной нагрузке?
Тестирование масштабируемости (Scalability Testing)	Как будет увеличиваться нагрузка на компоненты системы при увеличении числа пользователей?
Конфигурационное тестирование (Configuration Testing)	Какая будет производительность при изменении конфигурации системы?

Нагрузочное тестирование (load testing) является наиболее распространенным подвидом тестирования производительности. Данный вид тестирования проводится для оценки поведения системы при постоянном и неуклонном увеличении нагрузки до момента достижения порогового предела, который называют точкой насыщения. Точки насыщения — это уровень нагрузки, при котором дальнейшее наращивание числа пользователей ведёт к увеличению времени отклика системы либо ухудшению стабильности

системы, но не к увеличению в единицу времени количества полезных операций, обработанных системой. Основная цель нагрузочного тестирования состоит в определении максимальной нагрузки, которую может выдержать система, а также в выявлении «узких мест» приложения, влияющих на производительность[13].

График увеличения нагрузки на систему за некоторый период времени (профиль нагрузки) в случае нагрузочного тестирования представлен на рисунке 4:

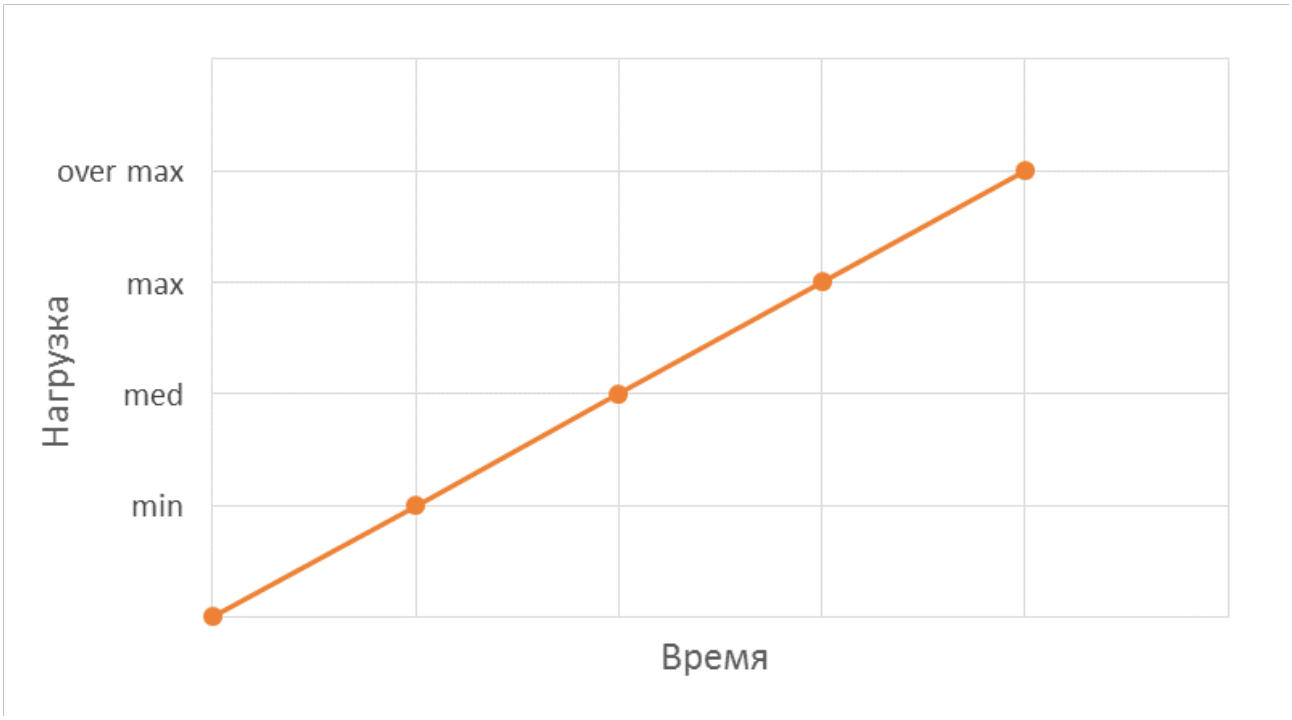

Рисунок 4. Профиль нагрузки для нагрузочного тестирования

При проведении нагрузочного тестирования общая производительность всей системы определяется следующими факторами [23]:

- скоростью работы программного обеспечения;

- скоростью работы аппаратного обеспечения;
- скоростью работы сети.

Атрибуты, которые отслеживаются для более точного измерения производительности и выявления «узких мест» при нагрузочном тестировании, включают в себя пиковую производительность, пропускную способность серверов, время выполнения критически важных транзакций при определенных уровнях нагрузки, максимальное количество одновременно работающих пользователей и границы приемлемой производительности при увеличении нагрузки.

Тестирование стабильности (stability testing) выполняется, чтобы убедиться, что программное обеспечение способно выдерживать ожидаемую нагрузку в течение длительного периода времени без какого-либо ухудшения времени отклика / пропускной способности[23]. Продолжительность прогонов определяются в соответствии с требованиями проекта. То есть в данном типе тестирования моделируется длительная работа пользователей для выявления перспективы долгосрочной беспрерывной работы системы, при этом сама нагрузка может иметь среднее значение (см. рисунок 5).

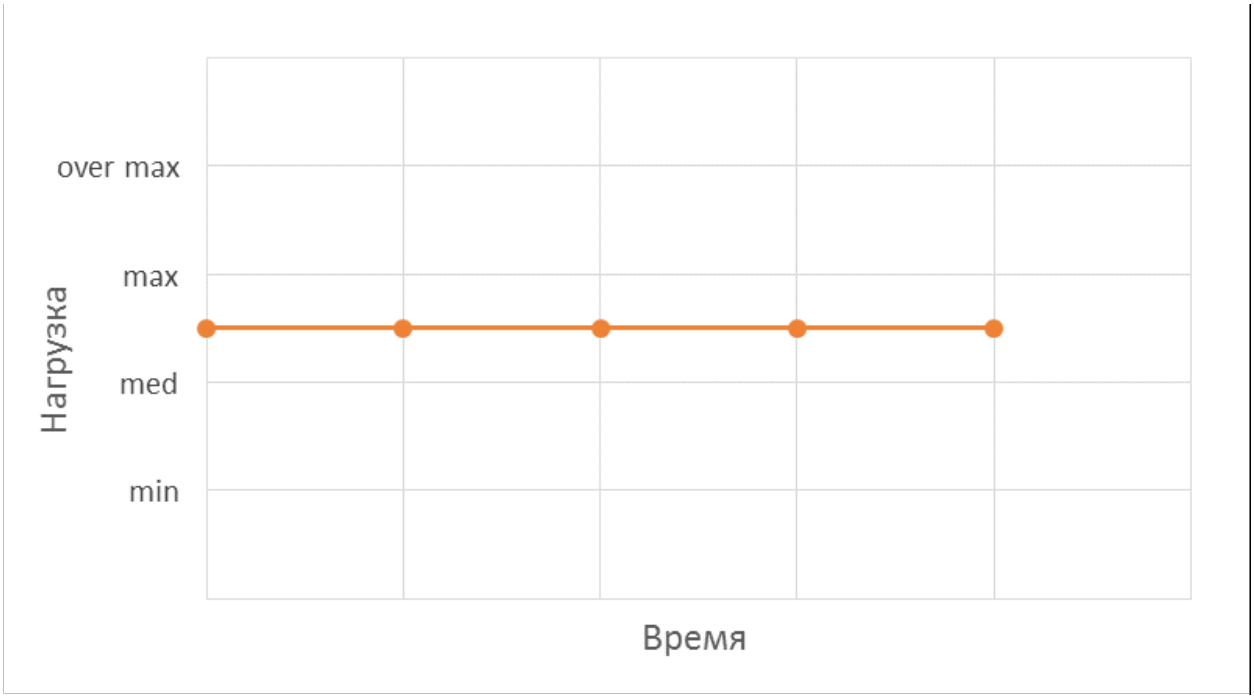

Рисунок 5. Профиль нагрузки для тестирования стабильности

Основная цель этого типа тестирования – выявить и предотвратить возможные утечки памяти, непредвиденные ограничения на количества выполнения некоторых транзакций, проблемы с оборудованием серверов (перезапуск сервером или ПО), отсутствие сбоев в работе сетевого оборудования через длительный период времени[27].

Стрессовое тестирование (Stress Testing) проводится для оценки производительности системы при работе под нагрузкой, близкой к пороговому значению, или превышающей ее. Стресс-тест создает нагрузку на аппаратные ресурсы, такие как ЦП, ОЗУ, диск, сеть и позволяет определить потенциальный предел приложения при некоторых ограниченных ресурсах.

При стрессовом тестировании на систему в течение небольшого периода времени подается огромная нагрузка, которая, после достижения некоторого значения, начинает снижаться (см. рисунок 6)

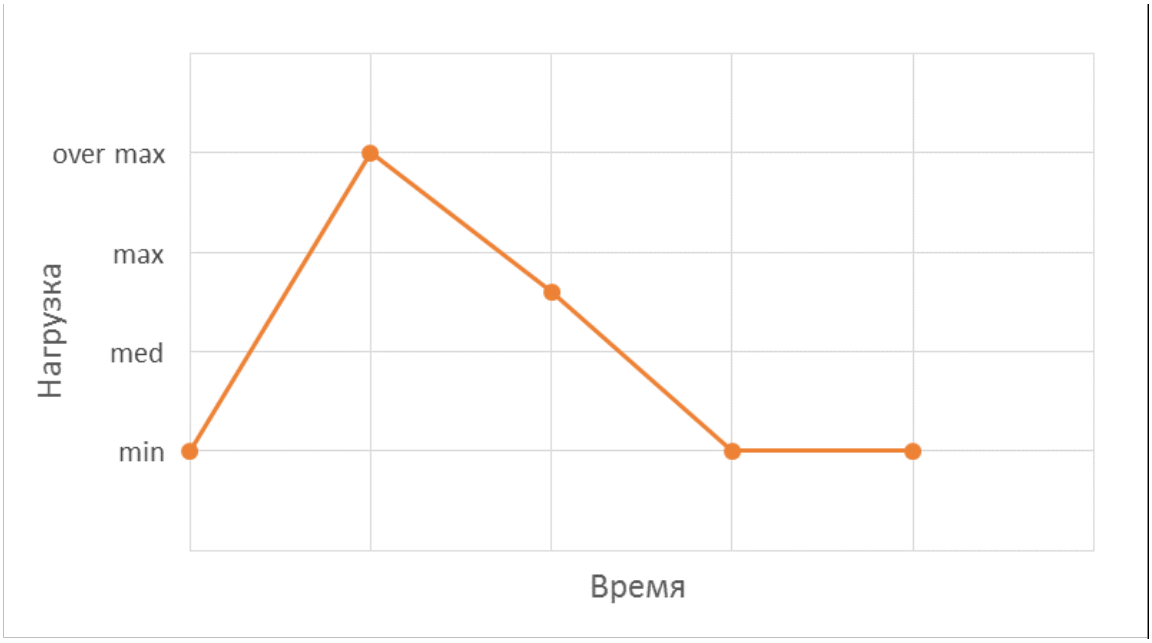

Рисунок 6. Профиль нагрузки для стресс-теста

Таким образом, можно выяснить, какие компоненты системы выйдут из строя первыми в экстремальных условиях, сможет ли система после этого восстановиться, проверить корректность логирования ошибок при большой нагрузке и своевременность оповещения о их возникновении.

Тестирование масштабируемости (Scalability Testing) проводится для определения эффективности работы приложения и проверки возможности его масштабирования под любым видом нагрузки. В ходе данного тестирования проверяются следующие виды масштабирования[5]:

- вертикальное масштабирование, при котором увеличивают производительности отдельных компонентов системы (добавление оперативной памяти, замена процессора или диска и т.д.);
- горизонтальное масштабирование, целью которого является распределение нагрузки на большее количество параллельно работающих серверов, выполняющих одни и те же функции;

Проведение **конфигурационного тестирования** (configuration testing) помогает определить производительность системы и ее отдельных компонентов на разных аппаратных и программных конфигурациях, убедиться что на конфигурациях программного и аппаратного обеспечения компоненты системы будут работать с одинаковой производительностью. А также выбрать оптимальную конфигурацию. При данном тестировании проверяется работа серверов на разных операционных системах, под управлением разных СУБД, если это web-приложение, то на разных браузерах.

Таки образом, конфигурационное тестирования представляет собой метод, который оценивает производительность программного обеспечения, аппаратных составляющих серверов с учетом различных конфигураций системы.

3.3 Процесс тестирования производительности

Методология, принятая для тестирования производительности, может широко варьироваться, но цель тестирования производительности всегда одна. проанализировав систему электронного бюджетирования, проведя исследование тестирования производительности, его видов и особенностей, была разработана методика для комплексного тестирования производительности информационных систем, планирования и анализа бюджета, которую можно представить в виде схемы, состоящей из 7 этапов (см. рисунок 7):

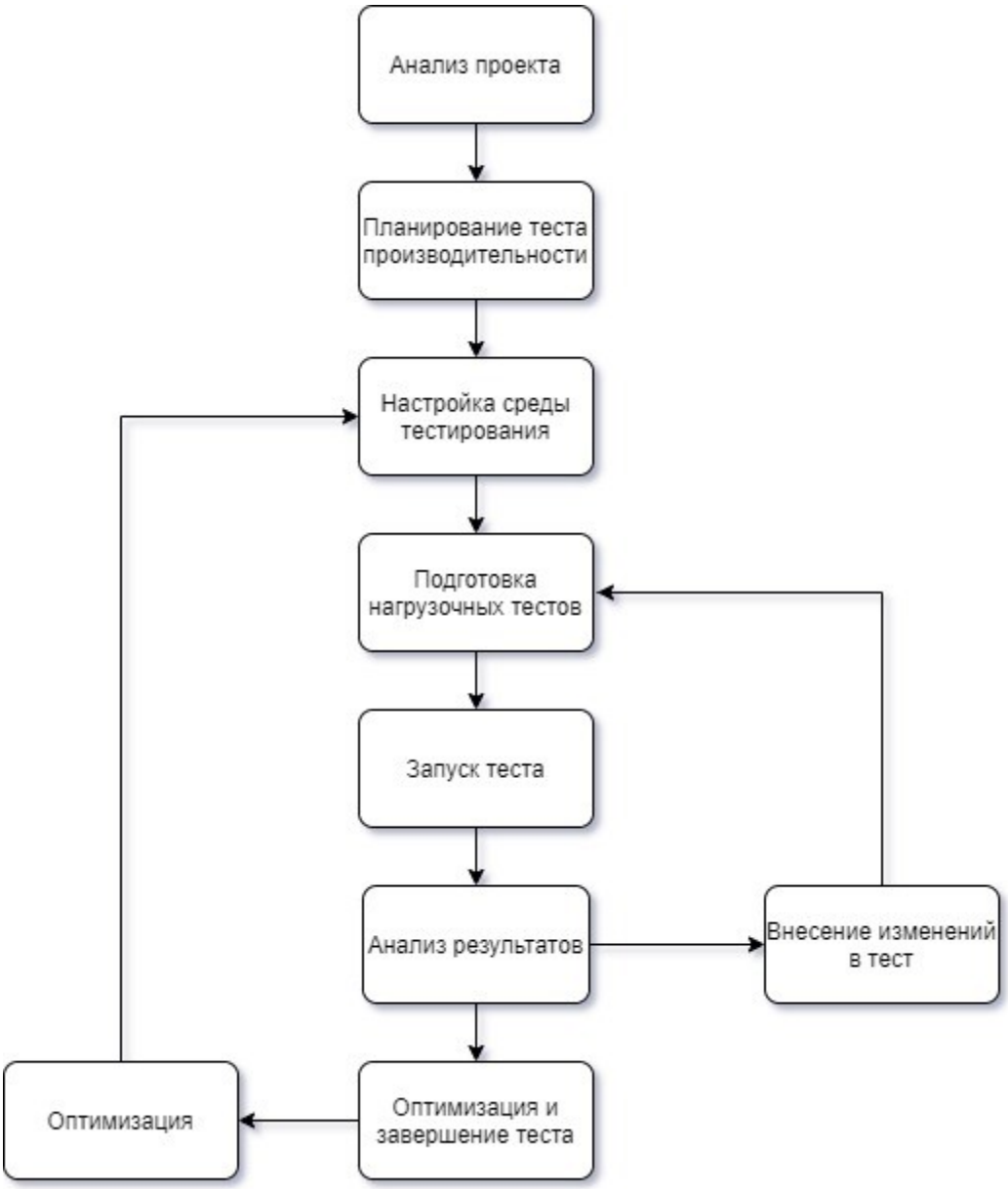

Рисунок 7. Схема тестирования производительности

- **Анализ проекта**. На данном этапе проводится большая работа по сбору информации о продукте, который необходимо протестировать. Изучается система и ее компоненты, подвергаемые нагрузочному тестированию, определяются нефункциональные требования к производительности системы (количество одновременно работающих пользователей, скорость ответа на запросы и т.д.). Проводится анализ функционала системы и на его основе формируются сценарии для нагрузочного тестирования. Так же выбирается инструмент для проведения нагрузочного тестирования.

- **Планирование теста производительности**. После того, как проведен анализ информационной системы, определены сценарии и требования к данным, следующим шагом будет описание видов и стратегий тестирования системы и формирование профиля нагрузки на их основе. Выбор стратегий тестирования будет в значительной степени определяться характером приложения и количеством времени, доступным для тестирования производительности.

- **Настройка среды тестирования**. Перед началом процесса тестирования следует ознакомится с деталями аппаратного и программного обеспечения. На этом этапе производится настройка серверов системы в соответствии с техническим заданием: настраивается аппаратная конфигурация серверов (количество и процессоров, оперативная память, выбирается система хранения данных, а

также пропускная способность сетевого канала), устанавливается и настраивается операционная система вместе с необходимым ПО. В итоге после всех установок и настроек производится пробный запуск системы и проверяется корректность работы всех компонентов информационной системы.

- **Подготовка нагрузочных тестов**. Имея сформированный профиль нагрузки, а также полностью настроенную среду тестирования можно приступать к созданию скрипта для нагрузочного тестирования. Скриптом для нагрузочного тестирования называют совокупность записанных запросов и заложенной логики выполнения бизнес-операций. В скрипт переносятся сценарии, сформированные на первом этапе, в соответствии со спецификой выбранного инструмента для НТ. Перед записью сценариев желательно выполнить их вручную на настроенном стенде, это позволит лучше ознакомиться с системой, которую предстоит протестировать, уменьшить время на запись сценариев и избежать неожиданных проблем. После внесения каждого сценарии в скрипт необходимо проверять его работу в тестовой среде. С помощью средств выбранного инструмента для проведения нагрузочного тестирования формируется профиль нагрузки.

- **Запуск тестов**. После создания нагрузочного скрипта и проверки корректности выполнения каждого сценария запускается многопоточный тест. Во время его выполнения выполняется контроль и мониторинг основных метрик тестирования производительности.

- **Анализ результатов**. На этом этапе собирается воедино вся полученная во время теста информация. После анализа может потребоваться внесение изменения в нагрузочный скрипт: изменение профиля нагрузки или отдельных сценариев в нагрузочном скрипте. После внесения всех изменений необходимо заново подготовить нагрузочный тест и его запуск.

- **Оптимизация и завершение тестирования**. После выполнения всех нагрузочных тестов формируется отчет о нагрузочном тестировании, в котором отображается поведение информационной системы под нагрузкой, перечисляются все найденные дефекты, а также формируются рекомендации к оптимизации программных компонентов. И после оптимизации и исправления недочетов необходимо заново произвести нагрузочное тестирование.

3.4 Инструменты для тестирования производительности, мониторинга и анализа результатов

Для проведения тестирования производительности необходимы специализированные инструменты. Главной задачей таких инструментов является генерация нагрузки на систему путем отправки запросов к серверу, тем самым имитируя работу реальных пользователей.

Существует множество инструментов для проведения нагрузочного тестирования. Выбор конкретного инструмента

зависит от многих факторов: поддержка инструментом определенного протокола передачи данных, стоимость лицензии, требования к оборудованию, поддержка платформы и многое другое[8].

Одним из самых простых инструментов для тестирования производительности HTTP-серверов является **ApacheBench** (AB). Это однопоточная программа, запускаемая из командной строки[31]. AB входит в состав Apache HTTP Server и подойдет для простого тестирования любого веб-сервиса. По окончанию тестирования AB выводит в консоль отчет, в котором выводится время выполнения запроса, количество переданной информации, перцентили и количество успешных запросов. С его помощью не получится провести сложное тестирование системы, состоящее из нескольких сценариев, поэтому его используют в основном для быстрой поверки производительности сервера.

Гораздо более продвинутым инструментом генерации нагрузки будет **Apache JMeter**. Это один из самых популярных и часто используемых инструментов генерации нагрузки. Этот инструмент развивается с 1999 года, имеет большой функционал и возможность интеграции с множеством сервисов[4]. Благодаря наличию графической оболочки, работать с этим инструментом не составляет особого труда, все расположено интуитивно понятно. Благодаря тому, что он написан на языке Java является платформо-независимым приложением, имеет встроенный proxy-сервер, что сильно упрощает и ускоряет запись сценариев, поддерживает многопоточную генерацию

нагрузки и большое количество протоколов передачи данных, среды которых HTTP, TCP, SOAP, FTP, LDPA и др.

JMeter имеет огромную пользовательскую базу, большое количество плагинов, расширяющих его функционал, а также возможность создания своих плагинов через API. Данный инструмент специально разрабатывался для использования специалистами по нагрузочному тестированию, выполняющим сложные, масштабные интеграционные нагрузочные тесты.

Однако из того факта, это Java приложение, разрабатываемое более двух десятков лет вытекает и основной его минус – относительно невысокая скорость работы по сравнению с современными инструментами для нагрузочного тестирования и необходимость выделения большого количества ресурсов для генерации нагрузки, особенно это касается потребления ОЗУ. Также JMeter имеет скудный функционал по построению отчетов о проведенном тестировании, но это решается подключением дополнительных инструментов. Но не смотря на все это, Apache JMeter идеально подходит для тестирования больших и сложных бизнес-приложений, написанных на Java[4].

Более современным инструментом является **Gatling -** очень мощный инструмент для нагрузочного тестирования, написанный на языке Scala. Он разработан для простоты использования, удобства обслуживания и достижения высокой нагрузки. Gatling построен на асинхронной архитектуре, что позволило реализовать генерацию виртуальных пользователей в качестве сообщений, а не

потоков[19]. Такой подход позволяет сэкономить генерировать достаточно большую нагрузку, задействую не так много аппаратных ресурсов.

Как и JMeter является кроссплатформенным приложением, поддерживает многопоточность. Из коробки Gatling умеет строить информативные графические отчеты, имеет совместимость с любыми платформами непрерывной интеграции, так как запуск теста происходит в командной строке. Также он поддерживает выгрузку метрик в InfluxDB, что позволяет сильно расширить возможности мониторинга и анализа результатов[19].

Однако данный инструмент не имеет графического интерфейса (только в платной корпоративной версии), а тесты создаются в виде исходного кода. А с выходом новых версий может сильно измениться API приложения, что создаст проблемы с обратной совместимостью ранее созданных тестов.

Tsung – это распределённая система нагрузочного и стресс-тестирования, написанная на языке Erlang. Tsung предлагает большой функционал для тестирования приложений по протоколу http, но он так же умеет работать и с многими другими протоколами. Благодаря тому что он написан на языке Erlang это, наверное, один из самых высокопроизводительных инструментов для нагрузочного тестирования, он отлично подходит для тестирования масштабируемости клиент-серверных приложений[26]. Так же имеет функционал для мониторинга аппаратных характеристик серверов, но не имеет собственного

графического интерфейса и подходит только для работы на операционных систем Linux/Unix.

Yandex.Tank — это расширяемый инструмент нагрузочного тестирования и анализа производительности веб-сервисов и приложений с открытым исходным кодом. В основе инструмента лежит высокопроизводительный асинхронный hit-based генератор HTTP-запросов phantom: он был переделан из одноименного веб-сервера, который «научили» работать в режиме клиента[25]. Yandex.Tank построен на модульной архитектуре, благодаря чему он может использовать в качестве генератора нагрузки Apache JMeter или Pandora, а собираемые метрики можно выгружать сразу в сервис Yandex.Overload для дальнейшего хранения и анализа результатов[22], или в базу данных InfluxDB для самостоятельного построения отчетов. Так же Yandex.Tank поддерживает настраиваемый и расширяемый мониторинг аппаратных характеристик серверов, работающий через SSH, которые так же выгружаются для построения детального отчета о проведенном нагрузочном тесте.

Однако недостаточно просто нагрузить систему, необходимо так же собрать большое количество данных во время теста и проанализировать их. Большинство инструментов для нагрузочного тестирования могут отслеживать основные метрики производительности (время ответа на запрос, процент успешно выполненных запросов, количество активных пользователей, количество запросов в единицу времени, пропускная способность сети и др.), часть из этих инструментов имеет встроенный функционал для построении отчетов на основании собранных данных, но для

качественного анализа производительности любой системы необходимо отслеживать также аппаратные характеристики серверов (использование процессора, оперативной памяти, нагрузка на диск и т.д.) и активность СУБД[6].

Для мониторинга потребления аппаратные характеристики серверов можно использовать встроенные инструменты операционной системы, но для дальнейшего анализа эти данные должны быть собранные. Для решения проблемы мониторинга аппаратные характеристик серверов существуют как платные, так и бесплатные решения. Главным отличием платных инструментов является относительная простота установки, настройки отдельных компонентов мониторинга, система помощи и поддержки. Однако в плане функциональных возможностей они почти не отличаются от программного обеспечения с открытым исходным кодом, поэтому они не рассматриваются.

Одной популярных систем мониторинга с открытым исходным кодом является **Zabbix**. Это продукт, предназначенный для администраторов крупных программно-аппаратных комплексов. Он подходит для мониторинга как крупных бизнес-систем, так и для средних и малых[34]. Zabbix достаточно прост в освоении, имеет обширный функционал и гибкую настройку (см. рисунок 8).

Рисунок 8. Интерфейс Zabbix

Для хранения метрик используются SQL базы данных, такие как MySQL, Oracle, PostgreSQL, SQLite и др. В систему уже встроен мониторинг стандартных метрик, среди которых:

- нагрузка на центральный процессор, в том числе отдельными процессами;
- использование оперативной памяти;
- нагрузка на жёсткий диск;
- объём свободной физической памяти;
- сетевая активность.

Zabbix может не только мониторить аппаратные характеристики серверов, но оповещать при отклонении метрик от нормы. Для этого используются специальные условия – триггеры[29]. В случае отсутствия функционала для мониторинга специфических метрик его можно написать самостоятельно с помощью API. Данный инструмент существует и развивается уже более 20 лет и имеет больше сообщество и множество дополнительных плагинов, среди которых имеются плагины для активности работы СУБД, но по информативности эти плагины не сравняться со специально предназначенными для этого инструментами.

Другим инструментом мониторинга активности серверов является **Grafana**. Это ПО во многом похоже на Zabbix, однако имеет ряд значительных отличий. Во-первых, Grafana представляет из себя средство для визуализации и анализа данных. Grafana позволяет создавать дашборды с панелями, каждая из которых отображает определенные показатели в течение установленного периода времени. Каждый дашборд универсален, поэтому его можно настроить для конкретного проекта или с учетом любых потребностей разработки и/или бизнеса. Также имеется большое количество уже готовых дашбордов, которые можно бесплатно скачать в сети Интернет.

Во-вторых, Grafana извлекает данные из базы данных временных рядов, которая специально оптимизирована для хранения метрик через пары времен и значений. В отличие от реляционных баз данных базы данных временных рядов используют индексацию данных, объединённых со временем, и, как следствие, скорость загрузки не уменьшается со

временем и остается достаточно стабильной (от 50 до 100 тыс. строк в секунду)[28].

Grafana включает встроенный анализатор запросов Graphite, который сильно упрощает процесс управления запросами. В Grafana встроена поддержка Prometheus - приложение, используемое для мониторинга событий и оповещений. А благодаря поддержке Elasticsearch извлекаемые данные можно сразу обрабатывать и сроить панели, отображающие среднее значение, минимальное/максимальное, перцентили, среднеквадратичное отклонение и многое другое.

Таким образом, Grafana является мощным инструментом для визуализации данных любым удобным способом. А при подключении дополнительных плагинов, Grafana может отображать метрики производительности (см. рисунок 9), но это поддерживают далеко не все инструменты для нагрузочного тестирования. Например, в Apache JMeter для подобной интеграции есть удобные плагины[20].

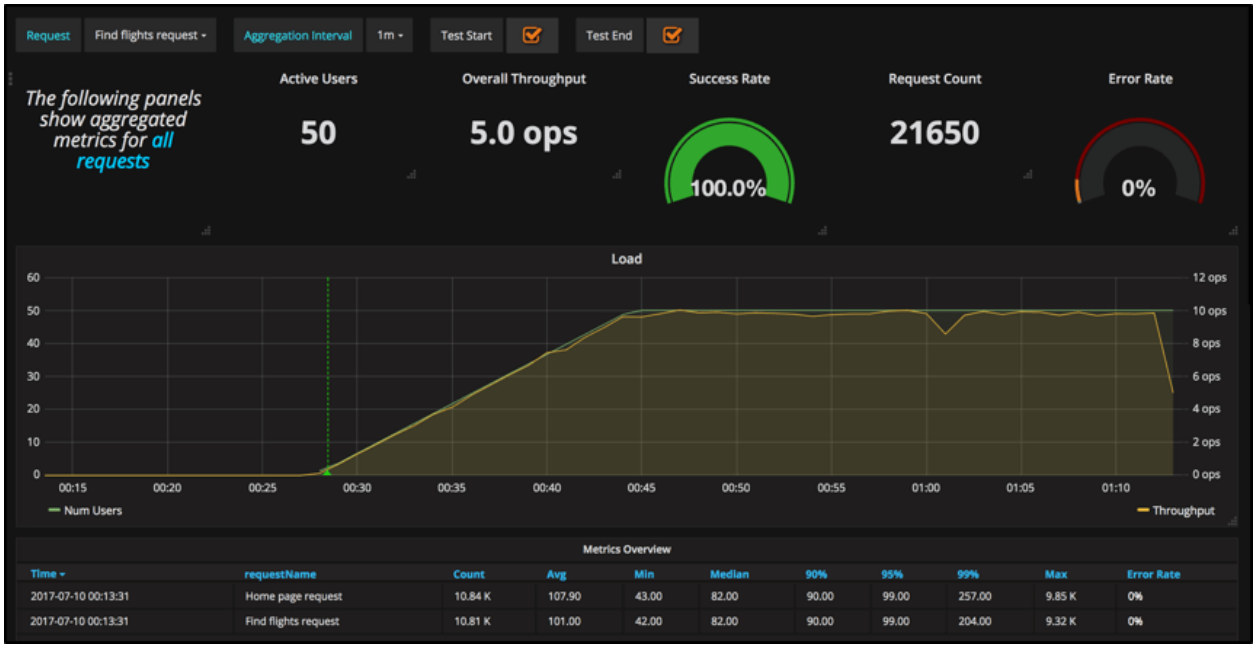

Рисунок 9. Мониторинг метрик производительности в Grafana

Хоть Grafana и может визуализировать метрики производительности, собираемые инструментом нагрузочного тестирования во время прогона теста, сбор аппаратных характеристик придется настраивать и производить отдельно, что создает сложности, которые можно избежать используя связку их нескольких инструментов[32]: Apache JMeter, Yandex.Tank и Yandex.Overload. Модульная архитектура Yandex.Tank позволяет использовать различные генераторы нагрузок, в том числе Apache JMeter. Во время выполнения тестового прогона Yandex.Tank собирает аппаратные метрики тестируемых серверов по протоколу SSH, выводит результаты в консоль (см. рисунок 10) и самое главное можно настроить отправку всех собираемых им данных в мониторинга и анализа производительности Yandex.Overload, которые отображает эти метрики в виде графиков и таблиц (см. рисунок 11).

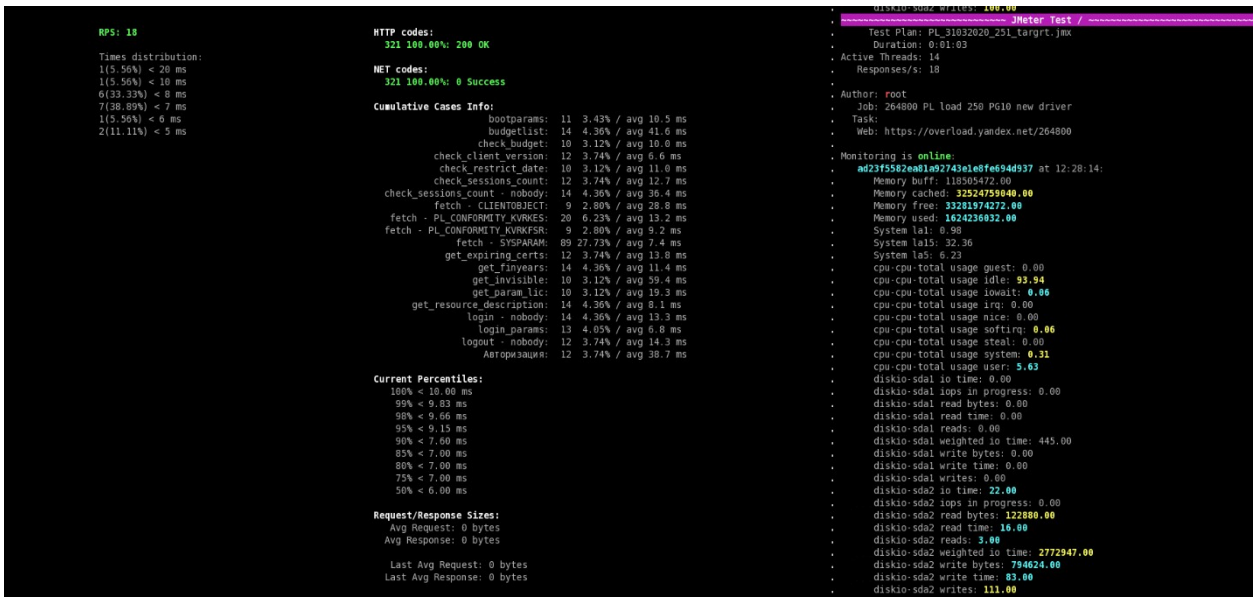

Рисунок 10. Консольный вывод метрик в Yandex.Tank

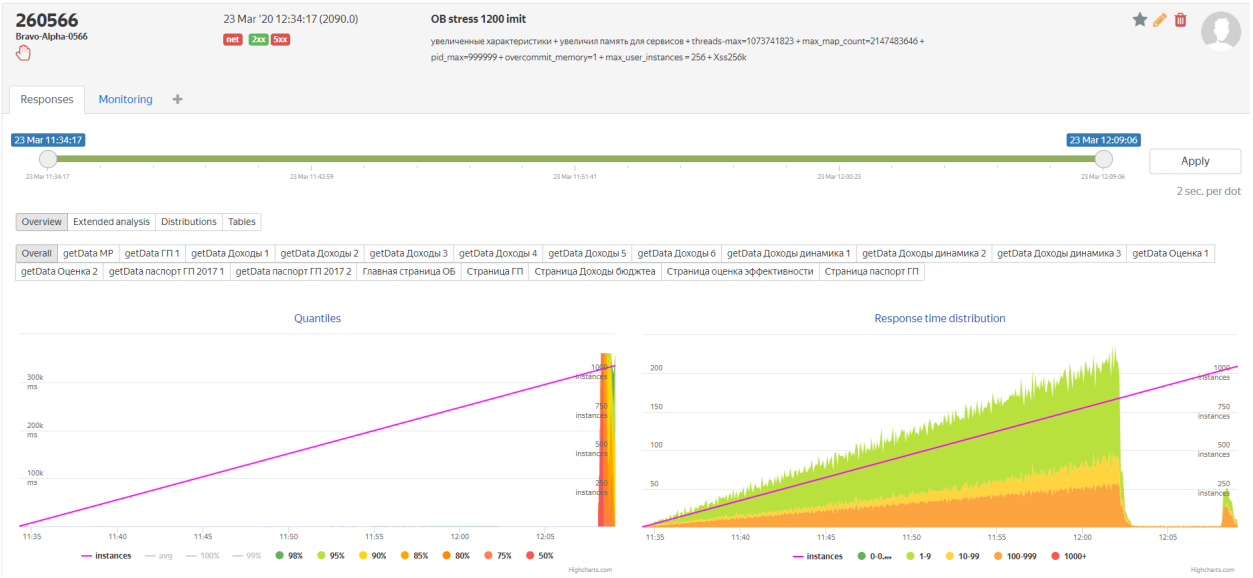

Рисунок 11. Отображение метрик в Yandex.Overload

Так же существуют инструменты, предназначение исключительно для мониторинга активности СУБД. Например, для Oracle это приложение **Oracle Enterprise Manager**. И хотя в первую очередь Oracle Enterprise Manager – это инструмент для управления программным и аппаратным обеспечение, разрабатываемым корпорацией Oracle[21], он имеет web-интерфейс для управления всеми функциями СУБД, а также для мониторинга текущего состояния системы. С его помощью можно отслеживать нагрузку на сервере, просматривать активные сессии, скорость выполнения запросов и потребляемые ими ресурсы, обнаружение блокировок данных и многое другое (см. рисунок 12).

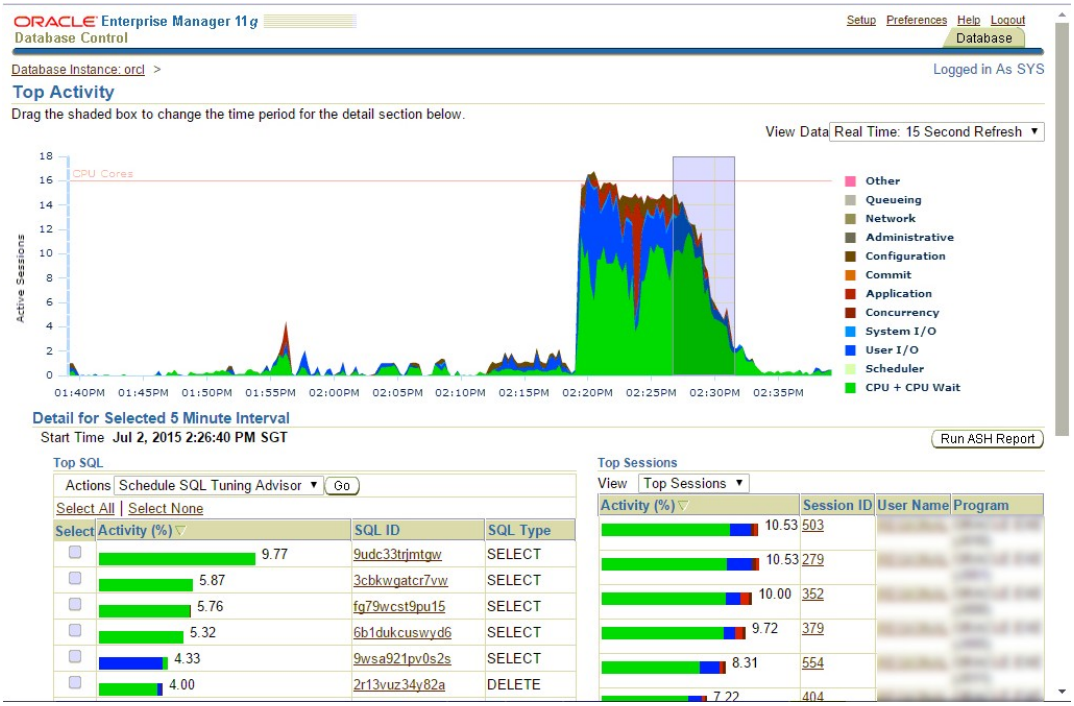

Рисунок 12. Вкладка производительности базы данных

Также в состав Oracle Enterprise Manager входит инструмент для создания детальных отчетов Automatic Workload Repository (AWR). AWR используется для сбора статистики производительности, включая[18]:

- Ожидание событий, используемых для выявления проблем с производительностью.
- Статистика временной модели, показывающая количество времени БД, связанного с процессом;
- Статистика активных сессий (ASH);
- Статистика использования объектов БД;
- Ресурсоемкие SQL операторы.

Для бесплатных СУБД, таких как PostgreSQL, имеется специализированное программное обеспечение с открытым исходным кодом для мониторинга и анализа активности и

производительности – **PoWA** (PostgreSQL Workload Analyzer). По сути, PoWA представляет из себя расширение, которое собирает статистику производительности и предоставляет ее в виде графиков в реальном времени[24]. С его помощью можно получить график выполнения запросов, блок хитов/прочтений, а также диаграмму затрат времени на выполнения запроса за выбранный период времени. Более того, при помощи POWA можно получить детальное описание по конкретному выполняемому запросу (см. рисунок 13).

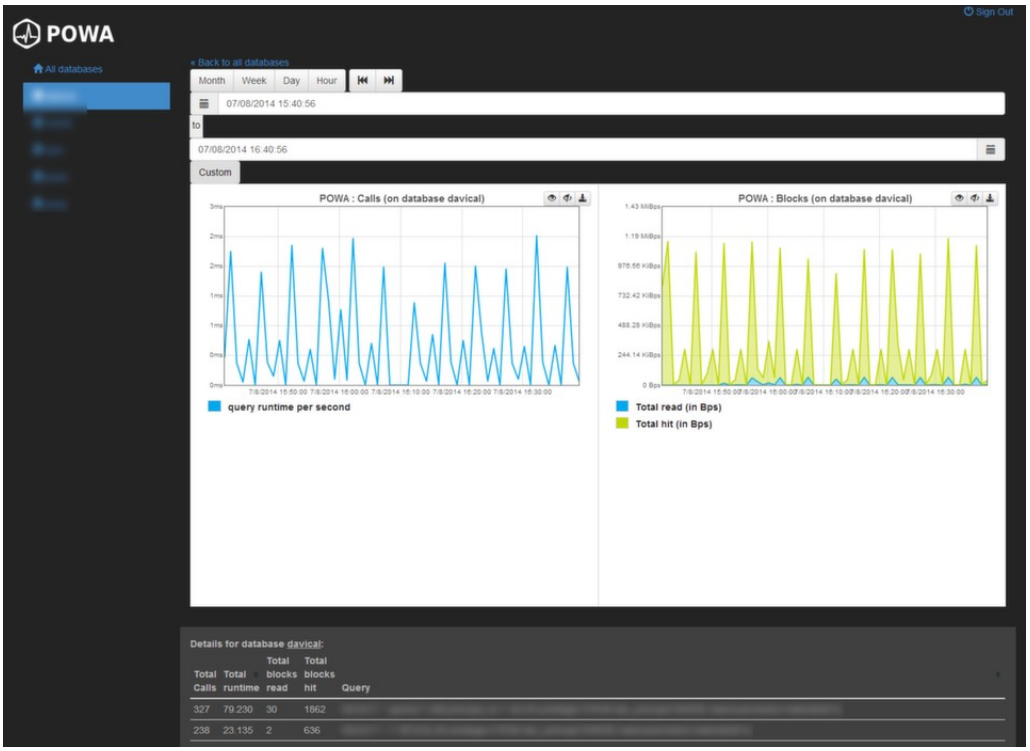

Рисунок 13. Web-интерфейс PoWA

Глава 3. Тестирование производительности системы «АЦК-Планирование»

3.1 Анализ проекта

«АЦК-Планирование» – это система автоматизации финансово-экономических органов, предназначенная для исполнения бюджета и управления бюджетным процессом в субъектах РФ и муниципальных образованиях.

Программный комплекс «АЦК-Планирование» предназначен для автоматизации процесса планирования бюджета региона (муниципального образования) на очередной финансовый год и плановый период в соответствии с действующим законодательством и новейшими тенденциями бюджетной реформы.

Система используется для автоматизации деятельности всех структурных подразделений финансового органа, включая его территориальные подразделения, участников бюджетного процесса, в том числе главных распорядителей бюджетных средств, распорядителей бюджетных средств, получателей бюджетных средств, бюджетных и автономных учреждений. Система предусматривает возможность информационного взаимодействия финансового органа с территориальными органами Федерального казначейства и кредитными организациями, осуществляющими кассовое обслуживание счетов бюджета, лицевых счетов участников бюджетного процесса, счетов по учету операций бюджетных

и автономных учреждений – лицевых счетов бюджетных и автономных учреждений соответственно (см. рисунок 14).

Рисунок 14. Схема управления планированием финансов

«АЦК-Планирование» имеет огромный функционал объединяет в себе множество опций, например:

- Формирование проекта бюджета;
- Расчет проекта бюджета;
- Формирование бюджета в проектном представлении;
- Планирование изменения в бюджете в течение года;
- Интеграция с ФНС;

- Финансово-экономический анализ параметров бюджета;
- Мониторинг исполнения бюджетных программ;
- Взаимодействие с государственной системой «Электронный бюджет»;
- Возможность работы в режиме удаленного доступа с использованием WEB-интерфейса.

В состав информационной системы «АЦК-Планирование» входят следующие элементы:

- система управления базой данных (СУБД);
- сервер приложений;
- система лицензионной защиты;
- клиентское приложение.

Такая схема обеспечивает равномерное распределение вычислительной нагрузки в локальной сети и хорошую масштабируемость системы в целом.

Для хранения, модификации и обработки информации используется СУБД производственного назначения, что гарантирует высокую скорость доступа к информации, ее целостность и непротиворечивость; дает возможность создания резервных копий данных и восстановления их в случае необходимости. К серверу БД предъявляются особые требования по производительности, безопасности и надежности.

Основная часть бизнес-логики обработки данных сконцентрирована на сервере приложений, что позволяет

централизованно хранить выполняемые процедуры, а также синхронизировать их работу в случае многопоточного доступа к данным. Сервер приложений системы «АЦК-Планирование» написан с использованием языка программирования Java и предназначен для выполнения в среде виртуальной машины Java (JRE/JDK). Тем самым обеспечивается его независимость от используемой операционной системы и высокая надежность в эксплуатации.

Он построен по типовой схеме монитора объектных транзакций (ОМТ) без запоминания промежуточного состояния и осуществляет обработку запросов клиентов и передачу этих запросов базе данных системы, а также получение данных из базы и передача их клиенту. Сервер приложения выступает промежуточным звеном между базой данных и клиентами, обеспечивая тем самым защиту данных и распределение нагрузки.

Доступ к данным, хранящимся в БД, осуществляется сервером приложений посредством драйверов JDBC, что обеспечивает независимость от используемой операционной системы и высокую надежность серверной части.

Клиентское приложение предназначено для подготовки и отправки заданий на сервер приложений, а также представления данных, полученных от сервера приложений, в доступной для восприятия конечным пользователем форме.

Система лицензионной защиты предназначена для предотвращения несанкционированного использования и

проверки подлинности модулей в процессе работы защищенной системы.

Также в состав системы дополнительно входят различные служебные программы (утилиты), использование которых носит вспомогательный характер. Например, для работы с системой «АЦК-Планирование» через WEB -интерфейс требуется WEB-сервер, который является промежуточным звеном между клиентами и сервером приложений, осуществляет обработку запросов клиентов и передачу этих запросов серверу приложений, а также получение данных от сервера приложений и передачу их клиентам.

В ходе тестирования должны выполняться бизнес-сценарии, оформляемые отдельными скриптами нагрузочного тестирования (см. приложение 1).

Для текущего тестирования производительности сервер лицензионной защиты можно исключить, так как он используется только в момент включения сервера приложения. Конфигурацию серверов приложения БД для нагрузочного тестирования и тестирования стабильности было решено выбрать для обеспечения одновременной работы 250 пользователей.

В результате анализа для информационной системы были предъявлены следующие требования к производительности тестируемого ПО:

1. Операции входа в систему не более 10 секунд для 95% случаев.
2. Сохранения документа не более 5 секунд для 95% случаев.

3. Время перевода статуса документа не более 5 секунд в 95% случаев.
4. Время вывода списка отфильтрованных документов не более 60 секунд.
5. Время вывода отчета не более 60 секунд

Для создания нагрузки со стороны пользователей на систему, будет использоваться связка из нескольких специализированных инструментов нагрузочного тестирования: Apache JMeter, Yandex.Tank и Yandex.Overload. Yandex.Tank будет использован в качестве инструмента для записи сценариев и генерации нагрузки на систему. Yandex.Tank будет выступать промежуточным звеном, основной задачей которого будет сбор аппаратных характеристик со всех нагружаемых серверов, сбор метрик производительности и отправка всех этих данных в сервис для мониторинга и анализа результатов тестирования Yandex.Overload. Мониторинг активности на сервере БД будет выполняться с помощью Oracle Enterprise Manager и PoWA для СУБД Oracle и PostgreSQL соответственно.

3.2 Планирование теста производительности

Для комплексного и всестороннего тестирования производительности системы «АЦК-Планирование» необходимо:

1. Провести нагрузочное тестирование;

2. На основании результатов нагрузочного тестирования сформировать профиль нагрузки для тестирования стабильности в течение 1-2 часов;
3. Проверить масштабируемость системы при работе с двумя и тремя серверами приложений;
4. Сравнить производительность при работе системы на СУБД PostgreSQL 10 и PostgreSQL 11 со старыми и новыми драйверами JDBC;

Для тестирования необходимо создать рабочий дамп базы данных, который будет восстанавливаться перед каждым запуском нагрузочных тестов, также перед каждым запуском необходимо производить сбор статистики. БД для тестирования должна содержать все данные, присутствующие в базе настоящей информационной системы. В нагрузочном скрипте настроить случайную задержку от 500 мс до 1000 мс между каждым запросом, а также задержку от 10000 мс до 15000 мс после выполнения каждого сценария там самым обеспечивая максимально возможное приближение к имитации работы реальных пользователей.

Для тестирования необходимо создать тестовых пользователей, которые будут привязаны к организациям. Всю нагрузку распределить между организациями как это указано в таблице 2:

Таблица 2. Распределение нагрузки

Наименование организации	Сценарии	% от общего числа пользователей
Минфин РК	Сценарий 1	36,03
	Сценарий 8	
	Сценарий 10	
	Сценарий 4	
	Сценарий 6	
Минэкономики Республики Коми	Сценарий 2	18,79
	Сценарий 7	
	Сценарий 9	
	Сценарий 11	
	Сценарий 6	
Минобрнауки Республики Коми	Сценарий 3	18,44
	Сценарий 5	
	Сценарий 9	
	Сценарий 4	
	Сценарий 6	
Минздрав Республики Коми	Сценарий 4	13,62
	Сценарий 3	
	Сценарий 6	
	Сценарий 5	
	Сценарий 10	
ГАУ РК "ЦИТ"	Сценарий 6	13,12
	Сценарий 4	
	Сценарий 9	
	Сценарий 2	
	Сценарий 7	

3.3 Настройка среды тестирования

Для выбранных стратегий тестирования понадобится создание и настройка шести серверов.

Для проведения нагрузочного, конфигурационного тестирования и тестирования стабильности потребуется один сервер приложения, один сервер БД и сервер генерации нагрузки. Для тестирования масштабируемости потребуется дополнительно еще два сервера приложения и отдельный сервер для балансировки нагрузки. Все сервера должны работать под управлением операционной системы Linux CentOS 6.8. На всех серверах должно быть установлено ПО Java 8 и серверный агент для сбора аппаратных метрик Telegraf.

Сервер БД для проведения нагрузочного тестирования и тестирования стабильности необходимо сконфигурировать для достижения одновременной работы 250 пользователей в соответствии с системными требованиями из технического задания: 2x4Core 3GHz Intel Xeon 64 bit, 32 Гб оперативной памяти и сетевое соединение со скоростью 2 Гбит. Для тестирования масштабируемости его аппаратные характеристики будет увеличены вдвое для одновременной работы в конфигурации с двумя и тремя серверами приложений. на данном сервере должны быть установленный объектно-реляционные СУБД Oracle 11.2.0.0, PostgreSQL 10.10 и PostgreSQL 11.5. Дополнительно для мониторинга и

анализа активности СУБД PostgreSQL необходимо установить инструмент PoWA.

На все серверы приложений должно быть установлено ПО, которые будет обрабатывать запросы пользователей, а также отправлять и принимать запросы от сервера БД. Конфигурация всех серверов приложений в отдельности настраивается для одновременной работы 250 пользователей в соответствии с системными требованиями из технического задания: 4x4Core 3GHz Intel Xeon 64 bit, 64 Гб оперативной памяти и сетевое соединение со скоростью от 2 до 4 Гбит. Дополнительно для сравнения производительности при работе системы на СУБД PostgreSQL 10 и PostgreSQL 11 со старыми и новыми драйверами JDBC, необходимо скачать свежие версии JDBC драйверов для СУБД PostgreSQL с официального сайта и переместить в папку с бинарными файлами приложения.

На сервер генерации нагрузки необходимо установить Apache JMeter и Yandex.Tank. Настроить SSH-соединение со всеми остальными серверами для мониторинга потребление аппаратных характеристик во время тестов через Telegraf. Так как это сервер, который будет имитировать работу одновременно большого количества пользователей с помощью Apache JMeter, для него следует выделить увеличенные ресурсы: 4x4Core 3GHz Intel Xeon 64 bit, 256 Гб оперативной памяти и сетевое соединение со скоростью 6 Гбит.

Последний сервер – это сервер балансировки нагрузки. Единственной его задачей будет принимать запросы от

сервера генерации нагрузки и распределять ее равномерно между серверами приложений. Для этого на сервер необходимо установить Nginx – бесплатное ПО с открытым исходным кодом, одной из функций которого является балансировка нагрузки для HTTP -серверов. Для работы этого сервера потребуется минимальная конфигурация: 1x2Core 3GHz Intel Xeon 64 bit, 8 Гб оперативной памяти и сетевое соединение со скоростью 1 Гбит.

После создания серверов, установки и настройки необходимого ПО надо проверить корректность взаимодействия серверов приложений и сервера БД, выполнить установку базы из дампа реальной информационной системы, при необходимости обновить эту базу и проверить работу SSH-соединения между серверами.

3.4 Подготовка нагрузочных тестов.

Имея рабочую среду тестирования, можно приступать к ознакомлению с функционалом информационной системы «АЦК-Планирование», выполнить вручную каждый сценарий в отдельности, понять, как работает система и какие контроли возникают при обработке электронных документов.

Нагрузочный скрипт Apache JMeter имеет туже структуру, что и файл формата XML. Главным элементом является «Test Plan». Внутри этого элемента уже содержатся все необходимые элементы для конфигурации, различные контроллеры, группы потоков и сами запросы (см. приложение 2).

Для записи сценариев Apache JMeter имеет встроенный proxy-сервер, с помощью которого можно легко перехватывать любые http-запросы. Главная задача в тестировании производительности это имитация работы пользователей. А в реальных условиях с системой работает много разных пользователей, и даже если они работают с одинаковым классом документов, данные внутри этих документов разные. Но при создании сценариев в тексте запроса только те данные, которые были введены в момент записи, и для того, чтобы сделать имитацию максимально приближенной к реальности необходимо обеспечить вариативность данных, иначе возникнут взаимные блокировки и долгое выполнение SQL-запросов на сервере БД. Для этого все записанные сценарии необходимо параметризировать, т.е. представить наиболее важные данные в виде переменных, которые будут выбираться из подготовленного массива данных. В качестве массива данных может выступать обычный csv файл, в который выгрузили некоторые данные из БД. Это могут быть имена пользователей и пароли, различные коды бюджетных классификаторов, идентификаторы записей и т.д. Для работы с такими данными в Apache JMeter есть специальный конфигурационный элемент «CSV Data Set Config» (см. рисунок 15):

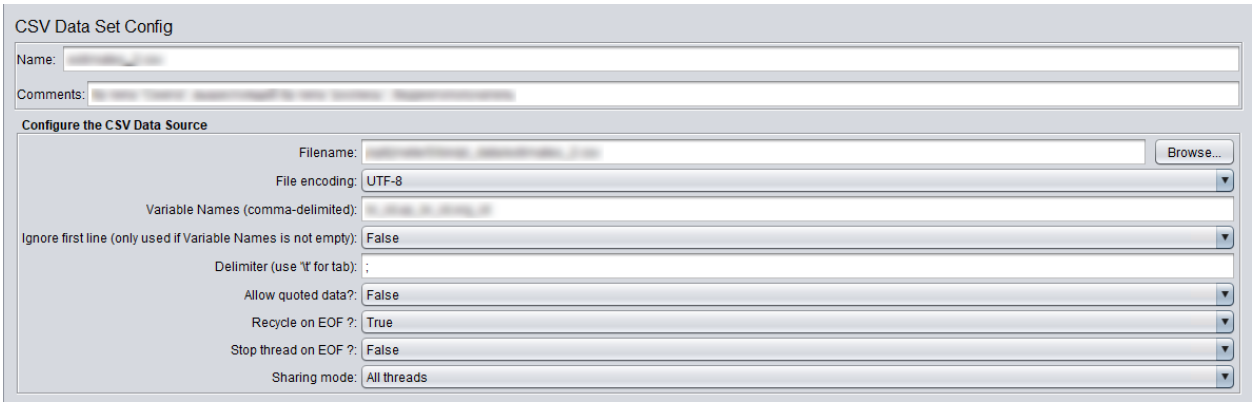

Рисунок 15. Элемент «CSV Data Set Config»

Так же некоторые переменные можно делать глобальными и давать им значения непосредственно в самом скрипте, для этого используется конфигурационный элемент «User Defined Variables» (см. рисунок 16):

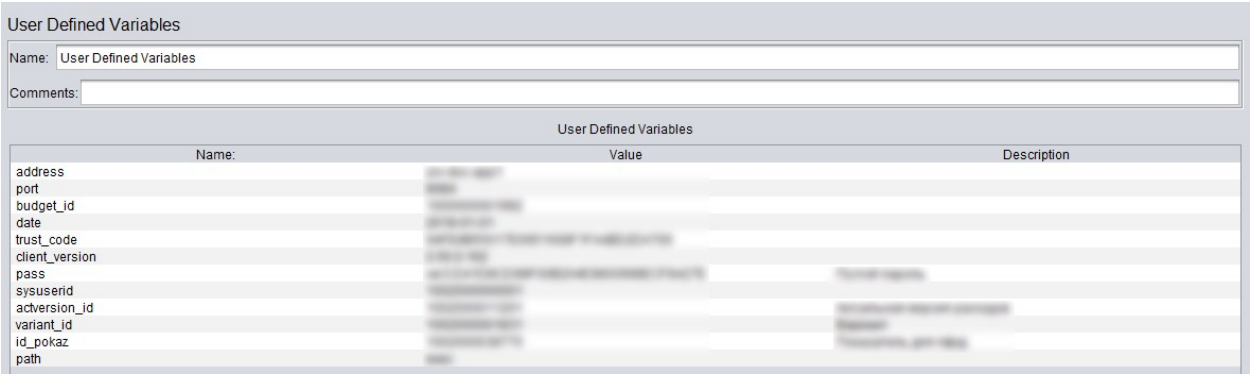

Рисунок 16. Элемент «User Defined Variables»

После выполнения запроса на создание нового элемента в системе «АЦК-Планирование» уникальный идентификатор генерируется автоматически непосредственно в базе данных и содержится в ответе на подобный запрос. Этот идентификатор необходим для дальнейших запросов, и чтобы его извлечь, записать в некоторую переменную и

использовать в дальнейшем в Jmeter предусмотренный экстракторы данных. Один из наиболее быстрых и эффективных экстракторов является «Regular Expression Extractor», которые извлекает данные по регулярному выражению и сразу записывает их в переменную(см. рисунок 17):

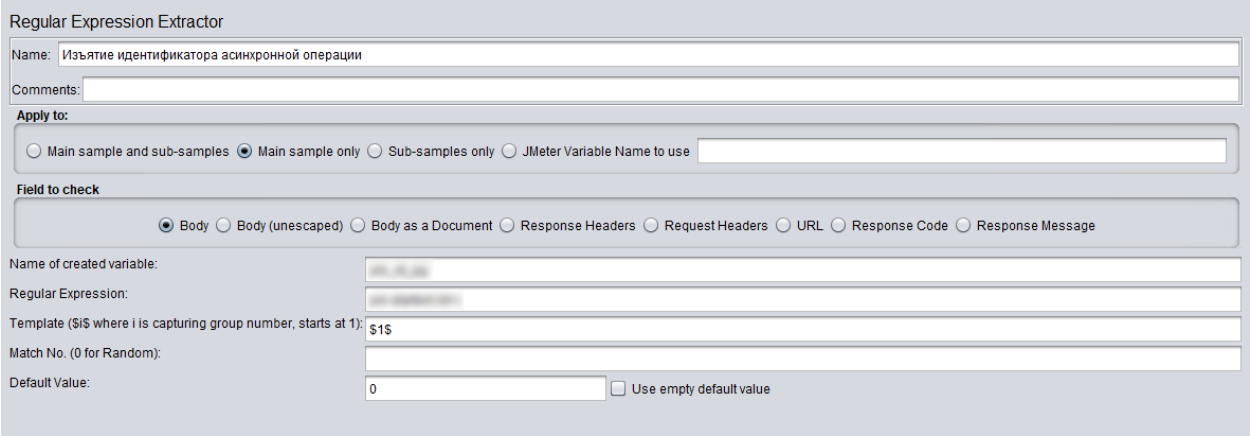

Рисунок 17. Элемент «Regular Expression Extractor»

Чтобы находить ошибки при работе виртуальных пользователей, в случае, когда http-запрос возвращается с кодом 200 (успешное выполнение), но фактически возвращается ответ с текстом ошибки сервера приложения или сервера БД, можно добавить элемент для проверки текста ответа «Response Assertion». Этот элемент можно настроить так, что он будет проверять возвращаемый http-запросом текст на наличие или отсутствие текстовых шаблонов. В случае тестирования системы «АЦК-Планирование» ответ на запрос не должен содержать текста «error:» (см. рисунок 18):

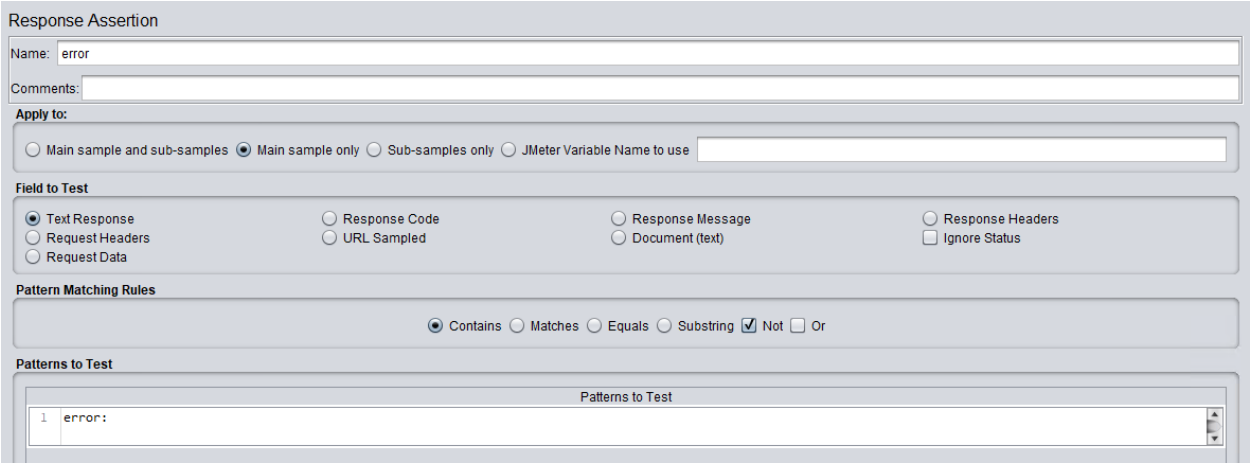

Рисунок 18. Элемент «Response Assertion»

Если во время тестов ошибки в работе системы находятся, то для удобства их можно сохранять в отдельные файлы с помощью элемента «Save Responses to a file», в котором также можно настроить сохранение ответа только ошибочных http-запросов, префикс к имени файлов и штамп временной метки как часть имени такого файла (см. рисунок 19):

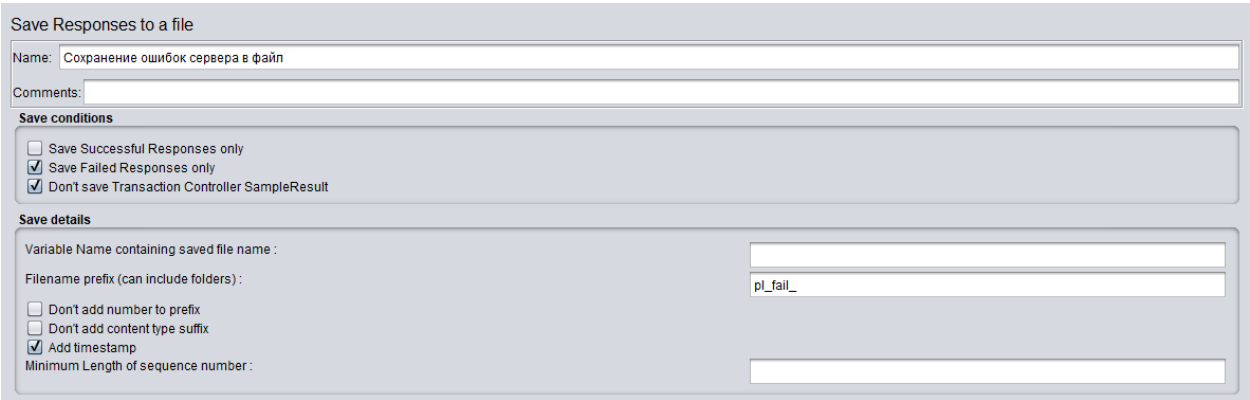

Рисунок 19. Элемент «Save Responses to a file»

Также в нагрузочном скрипте должны присутствовать случайные задержки между каждым http-запросом и после выхода пользователя из системы. Это реализуется с помощью

таймера «Uniform Random Timer», в котором настраивается постоянная задержка и максимальная случайная задержка в миллисекундах (см. рисунок 20):

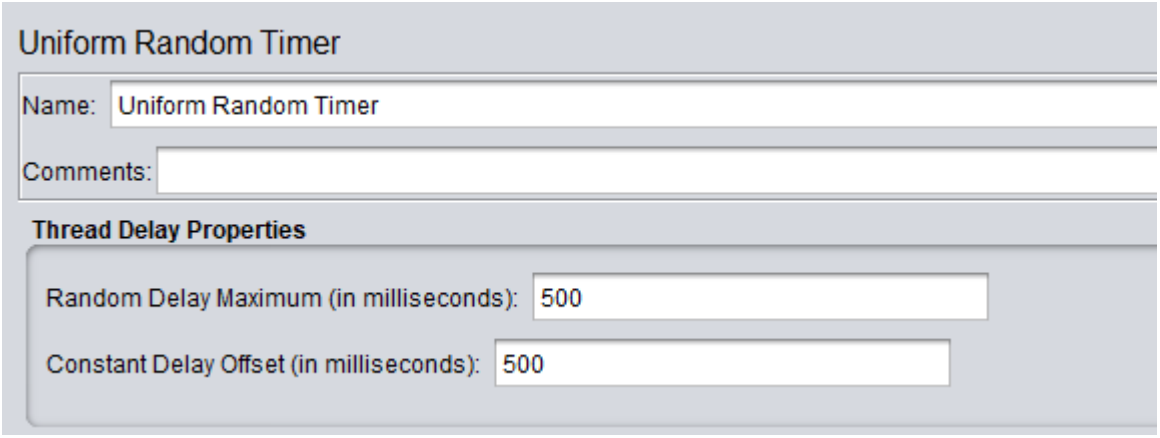

Рисунок 20. Элемент «Uniform Random Timer»

Так как скрипт нагрузочного теста имеет вложенную структуру, нет необходимости добавлять таймер к каждому http-запросу в отдельности, достаточно расположить этот конфигурационный элемент выше в иерархии, и он будет автоматически применяться ко всем запросам. Но для реализации задержки после выхода из системы, этот элемент необходимо поместить внутрь запроса.

Вследствие того, что сервер приложений системы «АЦК-Планирование» построен по схеме монитора объектных транзакций без запоминания промежуточного состояния, выполнение http-запросов на создание новых документов и их обработку происходит асинхронно. Т.е. в ответ на http-запрос на создание или обработку документа приходит не результат выполнение, а идентификатор данной задачи, по которому клиентское приложение отслеживает процесс выполнения данного запроса. И реализовать подобную специфическую

задачу в Apache JMeter можно решить с использованием логических контроллеров «If Controller» и «While Controller», в которых пишутся условия проверки статуса асинхронного задания и завершения цикла. А для измерения общего времени выполнения создания или обработки нового документа используется контроллер «transaction Controller», в который помещаются логические контроллеры и http-запросы (см. рисунок 21):

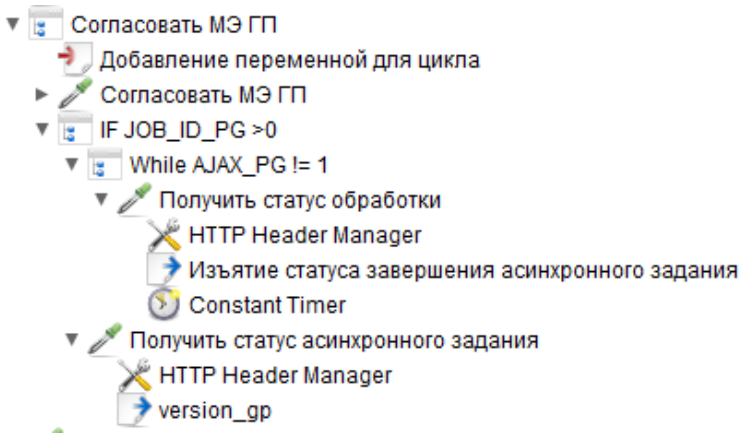

Рисунок 21. Реализация асинхронной обработки ЭД

Для распределения пользователей в соответствии с требованиями из таблицы 2 можно использовать контроллер пропускной способности «Throughput Controller». В данном контроллере настраивается количество виртуальных пользователей, которые выполнят вложенные в него сценарии в процентном соотношении (см. рисунок 22)

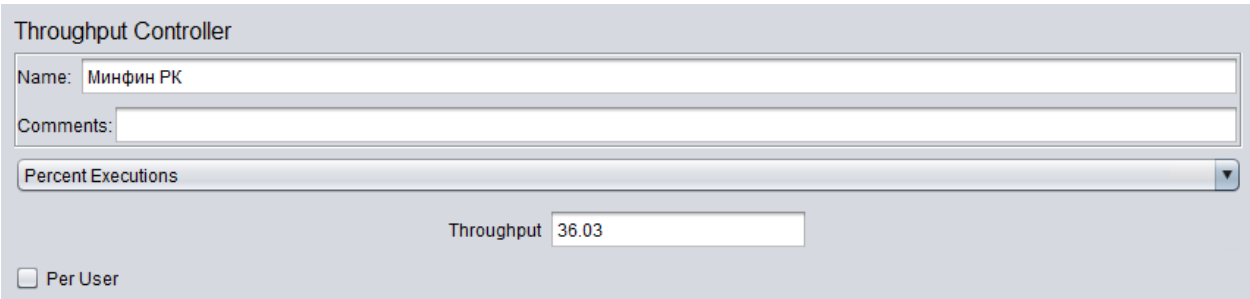

Рисунок 22. Элемент «Throughput Controller»

Таким образом, после записи всех сценариев, их проверки, обеспечения необходимой вариативности и настройки всех элементов, добавляется элемент, отвечающий за создание виртуальных пользователей – «Stepping Thread Group». В нем настраивается профиль нагрузки – количество виртуальных пользователей (потоков) и время выполнения нагрузочного теста в соответствии с типом проводимого тестирования производительности. В случае нагрузочного тестирования, виртуальные пользователи будут вводиться в систему методом линейного нарастания в течение 66 минут для достижения 1000 одновременно работающих пользователей (см. рисунок 23).

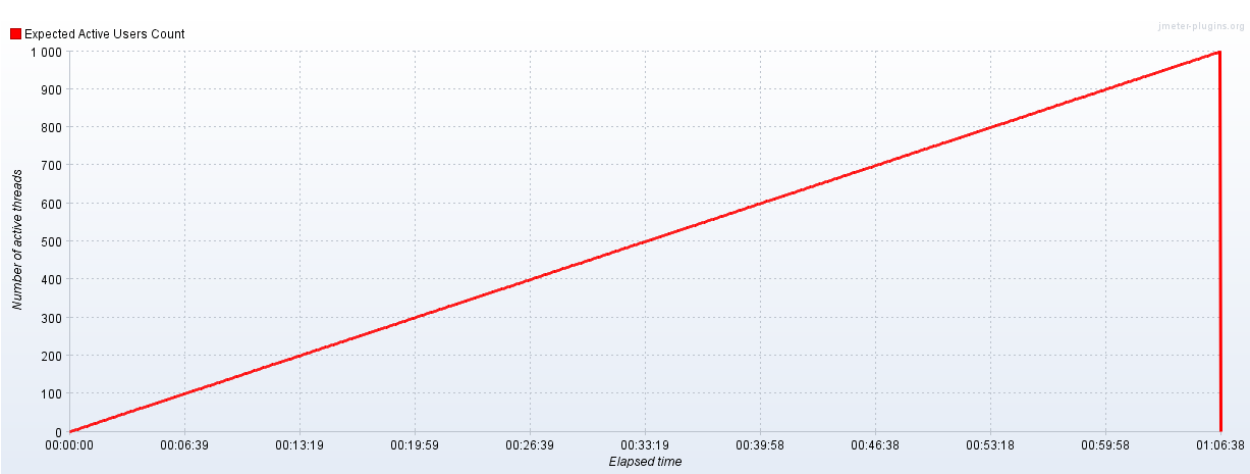

Рисунок 23. Профиль нагрузочного тестирования

В соответствии с конфигурацией серверов, для тестирования стабильности выбрана конфигурация нагрузки для достижения 250 одновременно работающих пользователей. Данное число пользователей будет введено в

систему методом линейного нарастания в течение 10 минут, и затем в течение 60 минут будет происходить непрерывная работа всех пользователей (см. рисунок 24). Такой же профиль будет применяться при конфигурационном тестировании.

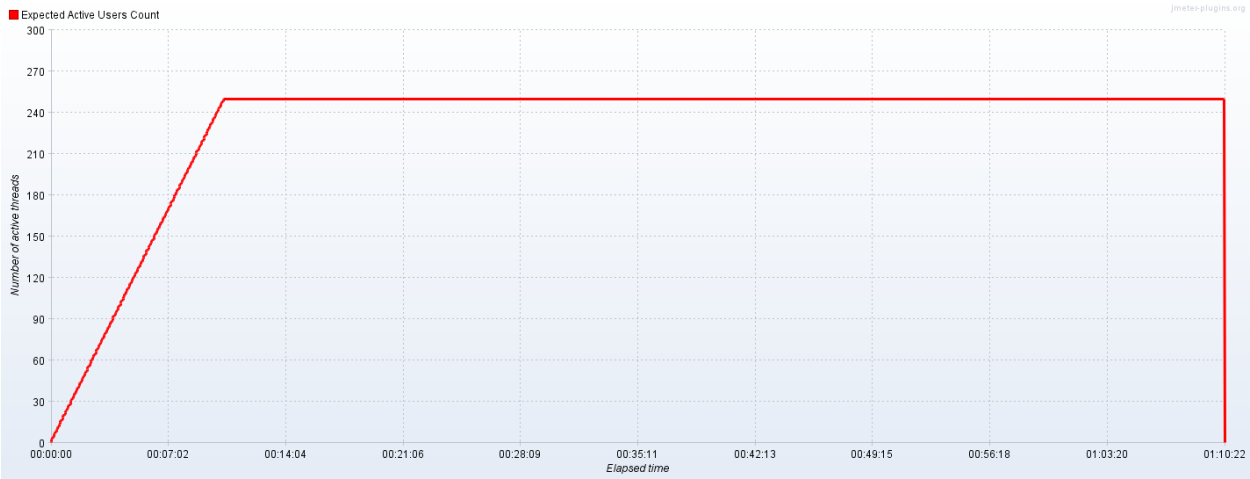

Рисунок 24. Профиль тестирования стабильности

Для тестирования масштабируемости профиль тестирования масштабируемости будет изменен для достижения 400 (см. рисунок 25) и 500 (см. рисунок 26) одновременно работающих на протяжении 120 минут для двух и трех серверов приложения соответственно.

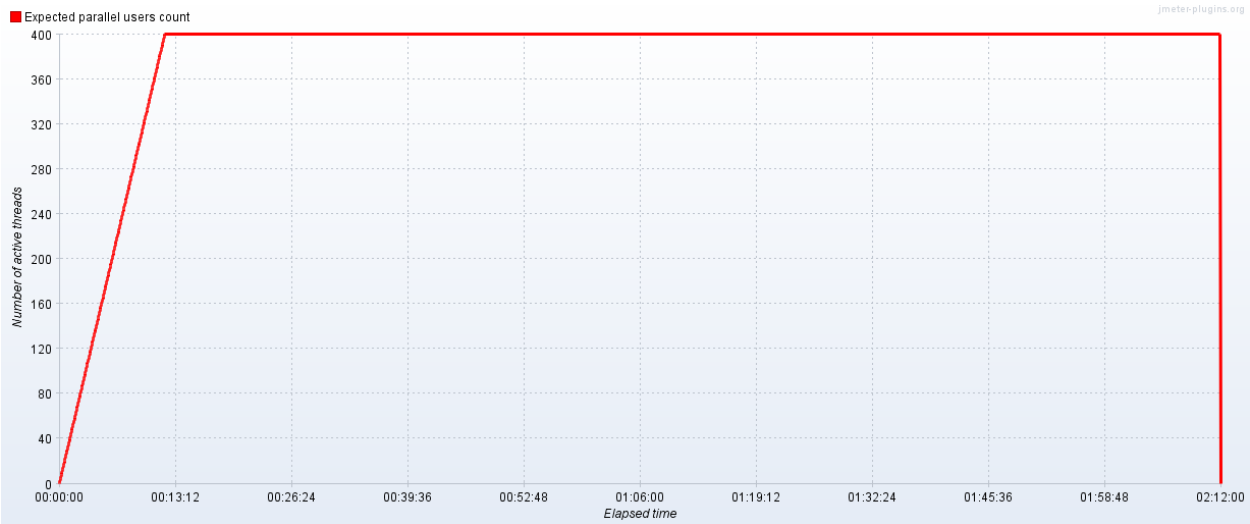

Рисунок 25. Профиль тестирования стабильности для двух серверов приложений

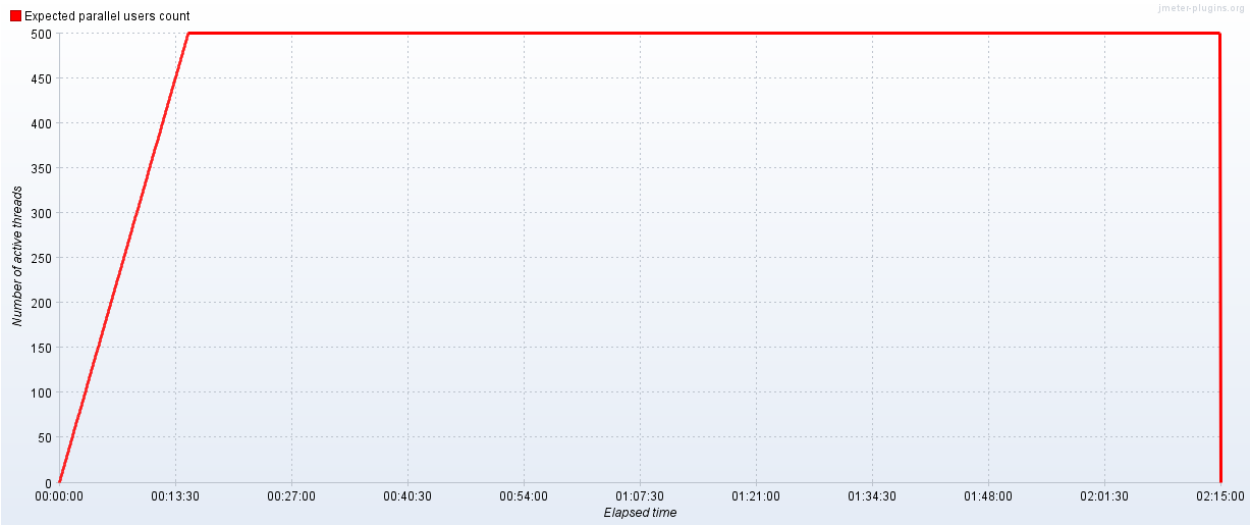

Рисунок 26. Профиль тестирования стабильности для трех серверов приложений

Таким образом, полностью готовый скрипт выглядит следующим образом (см. рисунок 27):

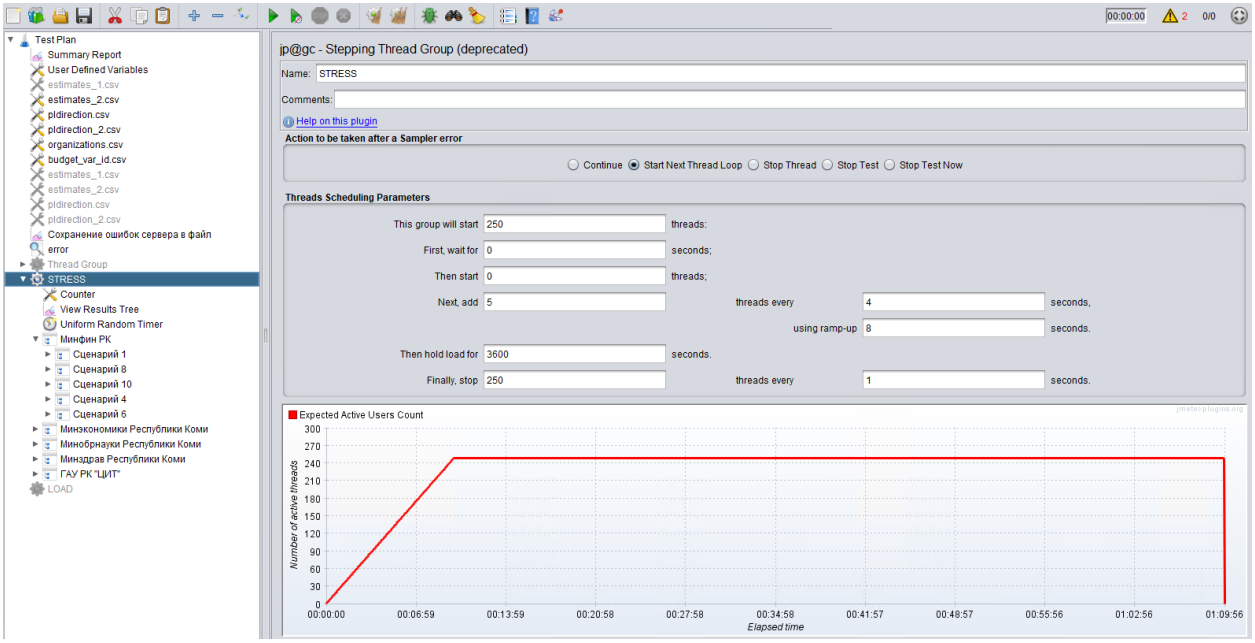

Рисунок 27. Готовый скрипт для нагрузочного тестирования

3.5 Запуск тестов.

После формирования и подготовки скрипта для нагрузочного тестирования следует приступать к тестированию системы. При тестировании производительности всегда выполняются следующие шаги:

1. Восстановление базы данных из ранее созданного дампа и сбор статистики;
2. Запуск серверов приложений;
3. Настройка сбора статистики на сервере генерации нагрузки;
4. Запуск теста производительности;
5. Мониторинг основных аппаратных ресурсов всех севров. А также метрик производительности на протяжении всего теста;
6. После завершения теста, сохранение логов на всех серверах;
7. Удаление базы данных.

3.6 Анализ результатов.

При первом запуске нагрузочного тестирования, система показала крайне низкую производительность (см. рисунок 28):

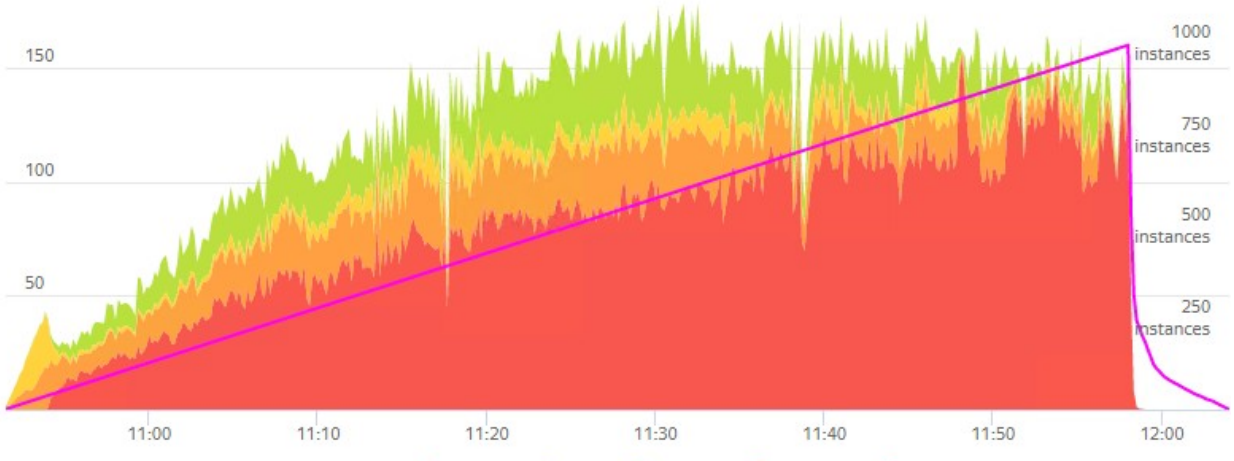

Рисунок 28. График производительности системы

На графике сиреневая линия отображает изменение количества виртуальных пользователей по времени. Зеленые, жёлтые, оранжевые и красные кривые отображают прирост транзакций в секунду.

После подключения 500 виртуальных пользователей количество HTTP-запросов в секунду (http-request per second - RPS) замерло на уровне 150. При детальном анализе активности БД был выявлен sql-запрос, который сильно нагружал сервер базы данных, вследствие чего все остальные запросы простаивали. Анализ логов СУБД помог выявить причину большой активности одного запроса, им оказался deadlock в данном запросе. После создания соответствующего дефекта с детальным описание причины низкой производительности, отделом разработки был оперативно выпущен патч, добавляющий возможность отключения механизма генерации номеров ЭД, который и вызывал проблемный sql-запрос.

После обновления сервера приложения, принудительного отключения механизма генерации номеров, производительность системы резко возросла (см. рисунок 29):

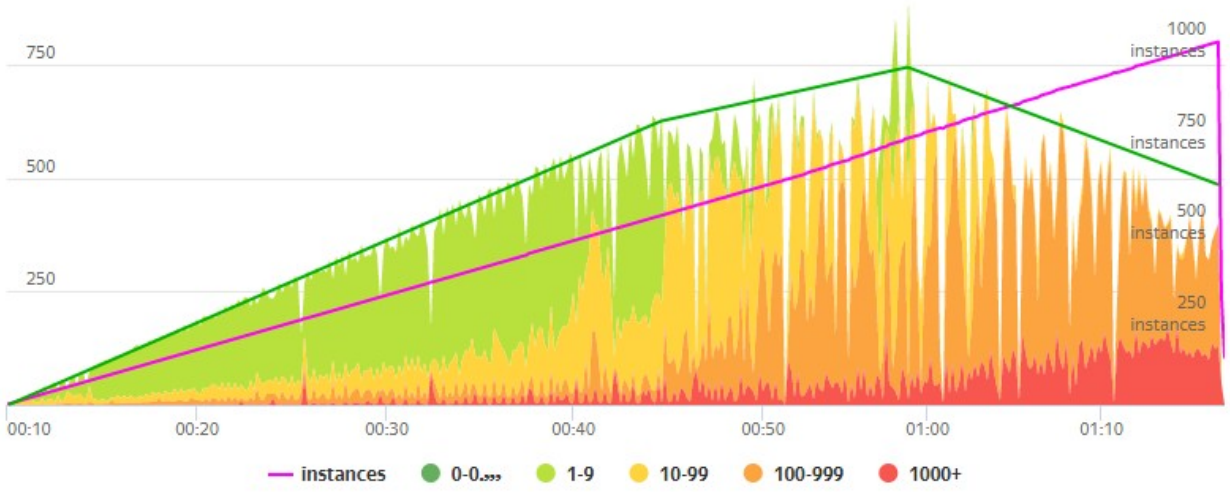

Рисунок 29. График производительности системы после патча

По итогам нагрузочного тестирования была определена точка насыщения, которая наступила в момент одновременной работы 734 пользователь со скоростью 890 RPS. После наступления точки насыщения рост производительности остановился, из-за того, что сервер БД был загружен выполнением большого количества запросов, направленных на построение форм списков, что привело к падению производительности и повышенной активности на процессор (см. рисунок 30):

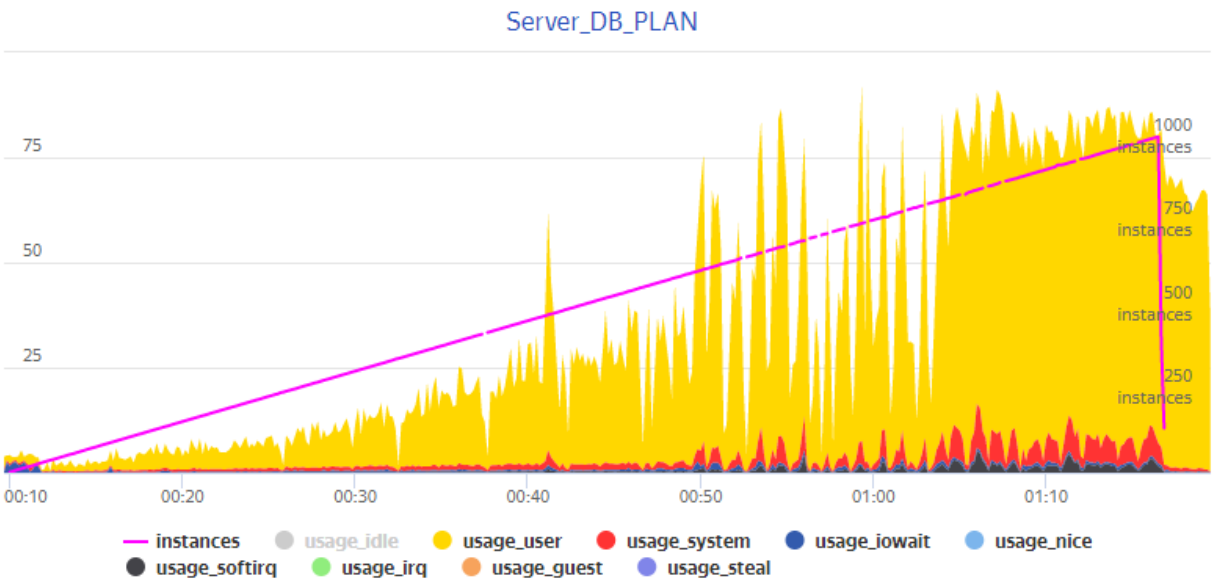

Рисунок 30. График нагрузки на процессор сервера БД

После нагрузочного теста был запущен тест стабильности, который проверил, подходит ли система под критерии производительности, заявленные в ТЗ. Результаты оказались положительными: система справилась с поставленной нагрузкой, показав равномерную производительность на протяжении всего теста (см. рисунок 31):

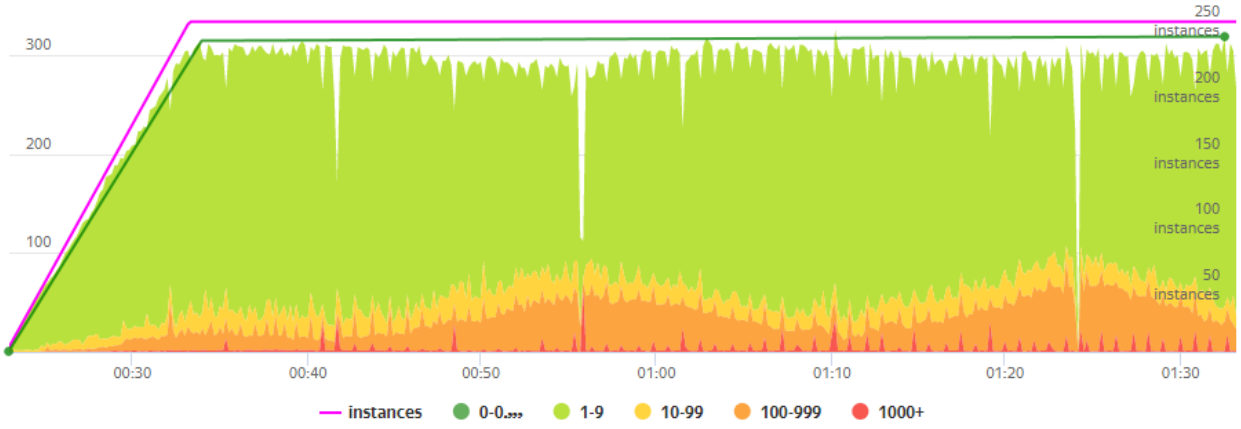

Рисунок 31. График производительности системы при тестировании стабильности

Следующим этапом тестировании производительности была проверка масштабируемости системы при работе с двумя и тремя серверами приложений. В соответствии с ТЗ, система, состоящая из двух серверов приложений и одного сервера БД, должна обеспечивать непрерывную работу до 400 пользователей с выполнением всех требования к производительности. В случае работы системы, состоящей из трех серверов приложений и одного сервера БД, должна обеспечиваться работа до 500 с выполнением тех же требований к производительности.

По результатам нагрузочного тестирования производительность системы при работе с двумя (см. рисунок 32) и тремя (см. рисунок 33) были найдены точки насыщения для каждой конфигурации:

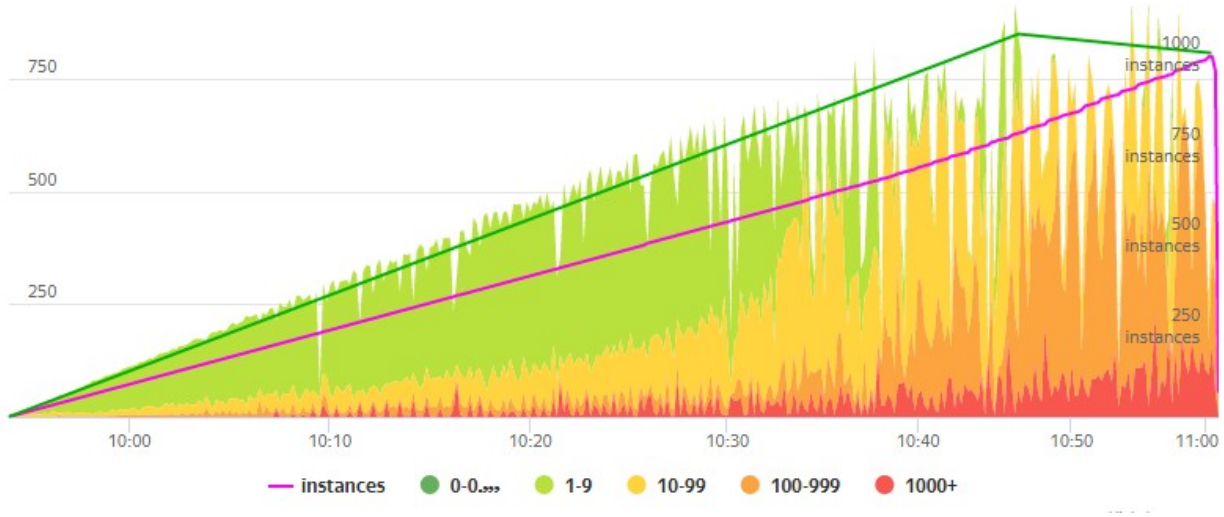

Рисунок 32. График производительности системы для нагрузочного теста при работе системы, состоящей из двух серверов приложений

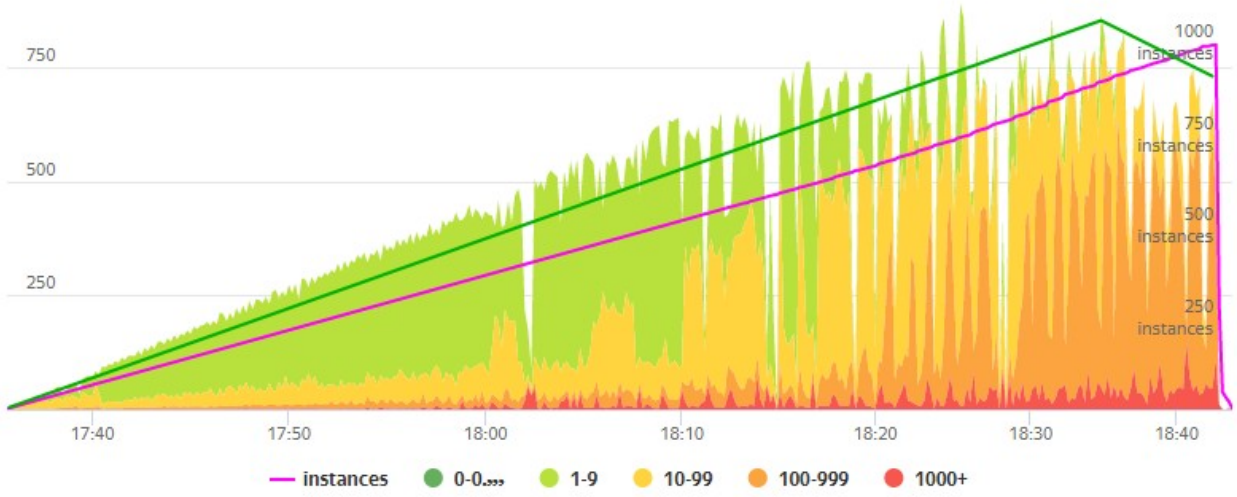

Рисунок 33. График производительности системы для нагрузочного теста при работе системы, состоящей из трех серверов приложений

В случае конфигурации системы из двух серверов приложений точка насыщения наступила при достижении одновременной работы 783 пользователь со скоростью 914 RPS. Для трех серверов приложений точка насыщения наступила при достижении одновременной работы 897 пользователь со скоростью 864 RPS.

После нагрузочного тестирования системы в требуемой конфигурации так же были выполнены прогоны тестирования стабильности для двух (см. рисунок 34) и для трех (см. рисунок 35) северов приложений:

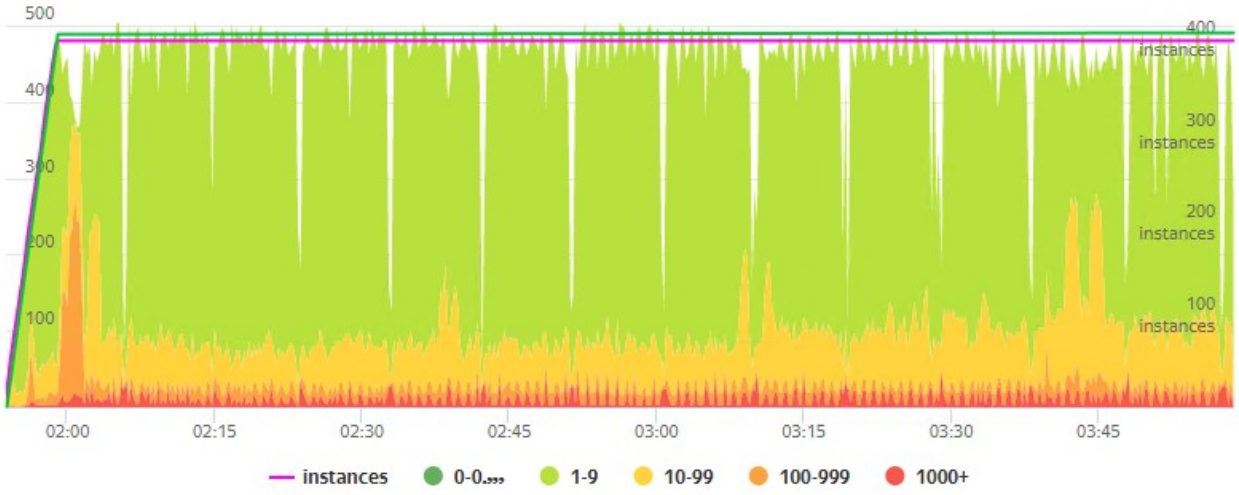

Рисунок 34. График производительности системы для теста стабильности при работе системы, состоящей из двух серверов приложений

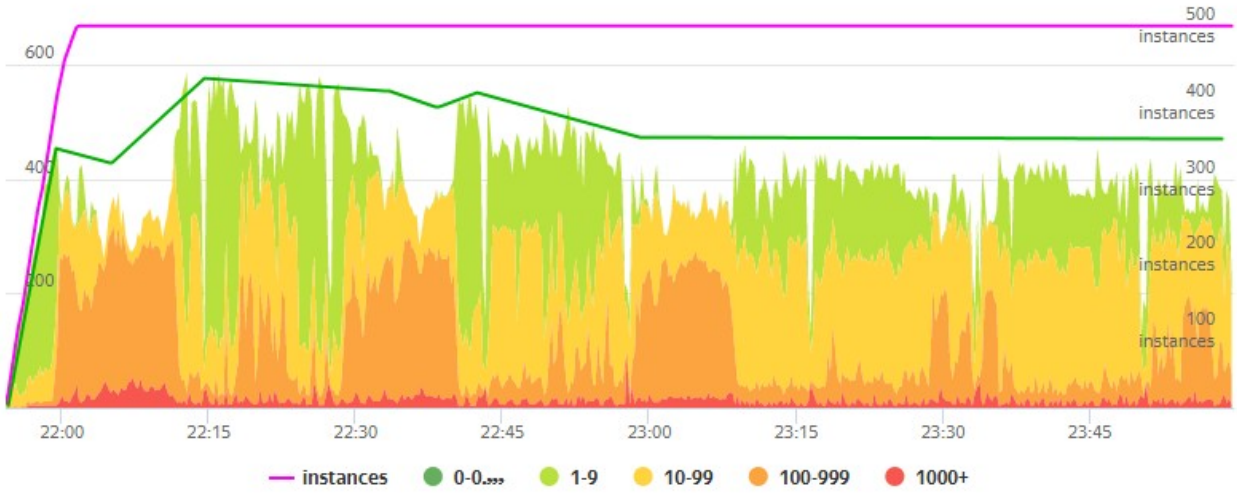

Рисунок 35. График производительности системы для теста стабильности при работе системы, состоящей из трех серверов приложений

Последней задачей было сравнение производительность при работе системы на СУБД PostgreSQL 10 и PostgreSQL 11 со старыми и новыми драйверами JDBC. Графики

производительности, полученные в ходе тестирования изображены на рисунках 36-39.

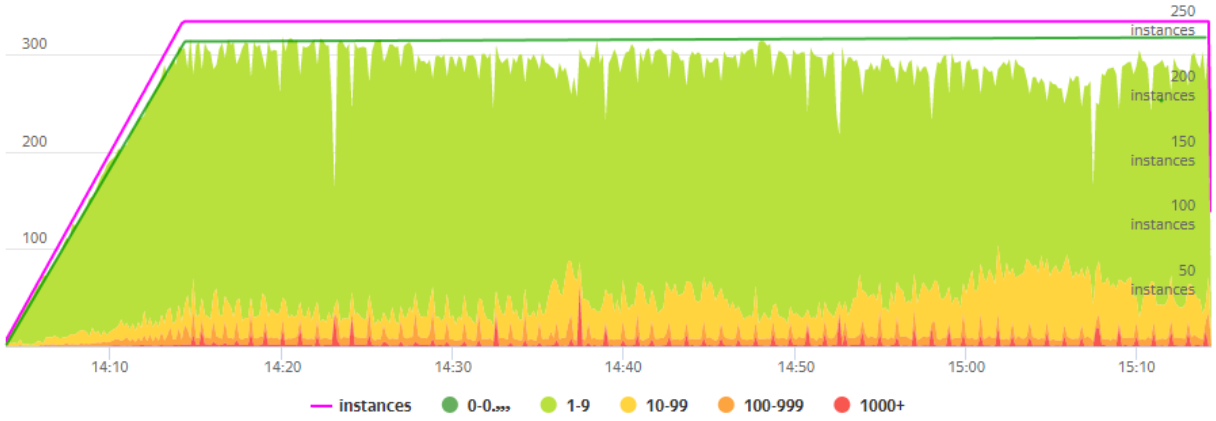

Рисунок 36. График производительности системы при работе на СУБД PostgreSQL 10 со старым драйвером JDBC

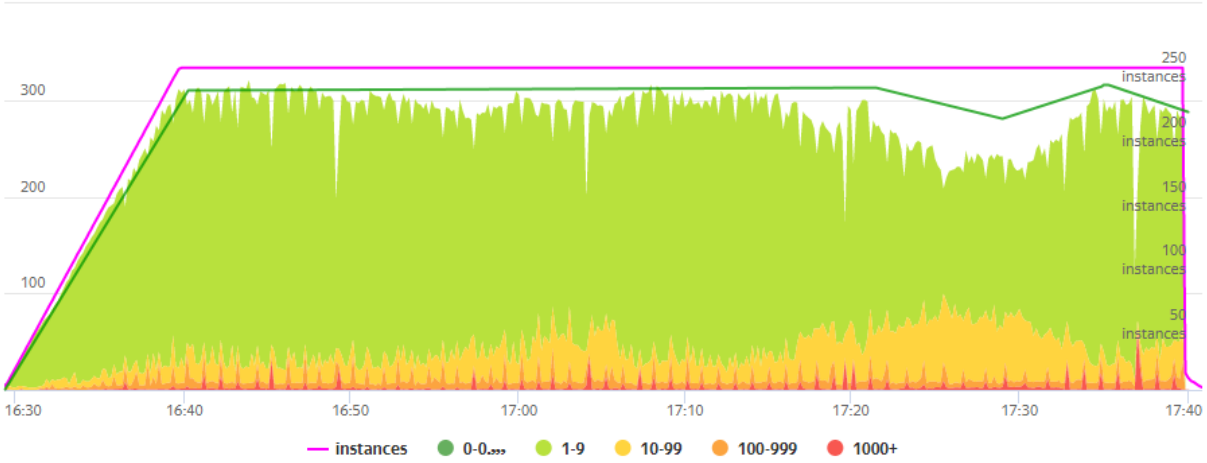

Рисунок 37. График производительности системы при работе на СУБД PostgreSQL 10 с новым драйвером JDBC

Рисунок 38. График производительности системы при работе на СУБД PostgreSQL 11 со старым драйвером JDBC

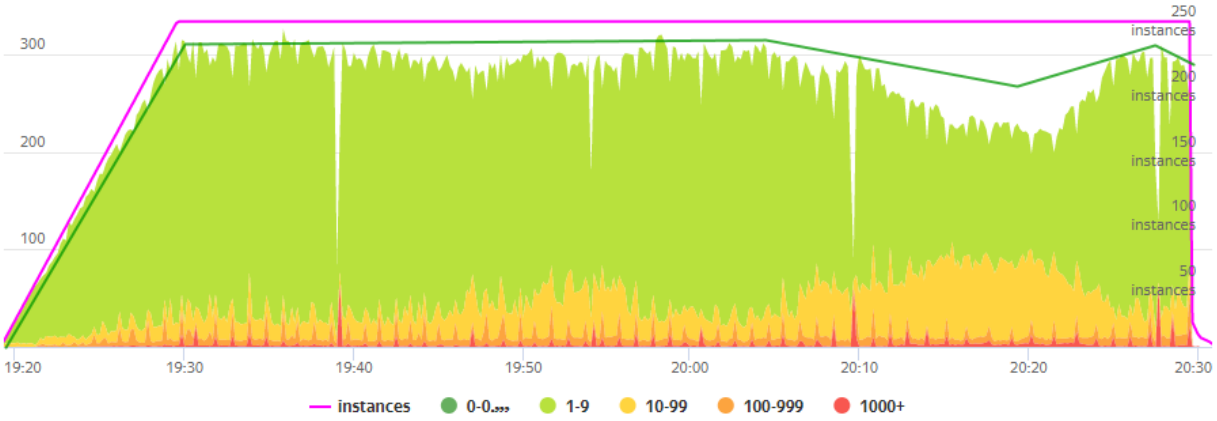

Рисунок 35. График производительности системы при работе на СУБД PostgreSQL 11 с новым драйвером JDBC

3.7 Оптимизация и завершение тестирования

Таким образом, при тестировании системы в конфигурации 1 сервер приложения и 1 сервер БД был обнаружен блокирующий запрос в механизме генерации номеров для ЭД. Но при принудительном отключении этого

механизма, система справляется с поставленной нагрузкой и способна обеспечить комфортную работу 250 и более пользователей при штатном режиме нагрузки (обычная работа) со временем доступа к данным менее 2 секунд, как заявлено в технических требованиях, а также имеет большой запас производительности для работы во время пиковой нагрузки.

Тестирование системы в конфигурации 2 сервера приложений так же показала неплохую производительность, соответствующая заявленным требованиям. Однако при тестировании стабильности системы в конфигурации трех серверов приложений производительность была неравномерной, на графике заметны провалы, связанные с повышенной активностью на сервере БД. Это свидетельствует о плохой оптимизации программного кода, вследствие чего время обработки некоторых классов ЭД сильно возрастает и превышает максимально допустимое время, указанное в требованиях. Также во время проведения конфигурационного тестирования балы замечена периодическая просадка в производительности, и аномальная нагрузка на сервер БД. Многопоточное тестирование каждого сценария в отдельности выявило еще 2 взаимные блокировки.

Сравнение производительность при работе системы на СУБД PostgreSQL 10 и PostgreSQL 11 со старыми и новыми драйверами JDBC не показало улучшений. Но при работе с новыми драйверами были обнаружены ошибки, связанные с отсутствием некоторых методов в новой версии драйверов. Для получения полноценного результата следует

предварительно оптимизировать приложение под работу с новыми драйверами JDBC.

Заключение

В ходе выполнения выпускной квалификационной работы была разработана схема схемы комплексного тестирования производительности информационной системы, предназначенной для планирования и анализа бюджета.

Для достижения цели были выполнены следующие задачи, сформулированные во введении:

- Изучены особенности электронного бюджетирования РФ;
- Исследованы инструменты для тестирования производительности и анализа результатов тестирования;
- Разработана схема и проведено комплексное тестирование производительности системы «АЦК-Планирование»;
- Проанализированы результаты тестирования производительности.

Перед разработкой схемы была исследована бюджетная система РФ, система электронного бюджетирования и особенности ее использование, изучена концепция производительности и исследованы основные виды тестирования производительности.

Следуя разработанной схеме, было проведено комплексное тестирование производительности программного комплекса «АЦК-Планирование», в результате которого были обнаружены и исправлены критические уязвимости в производительности системы.

Разработанная схема успешно применяется в компании «БФТ» и показывает свою эффективность не только на информационных системах, предназначенной для планирования и анализа бюджета, но и на многих друг, в том числе не относящимся к информационным системам финансового управления.

Результаты выпускной квалификационной работы были апробированы в виде научной стать, опубликованной в международном научном журнале «Молодой ученый» №18 (308), май 2020 г. (см. приложение 3).

Список использованной литературы

1. Федеральный закон от 29 июня 2015 г. № 188-ФЗ «О внесении изменений в Федеральный закон "Об информации, информационных технологиях и о защите информации" и статью 14 Федерального закона "О контрактной системе в сфере закупок товаров, работ, услуг для обеспечения государственных и муниципальных нужд» // Информационно-справочная система «Консультант Плюс».

2. Постановление Правительства РФ от 16 ноября 2015 г. № 1236 «Об установлении запрета на допуск программного обеспечения, происходящего из иностранных государств, для целей осуществления закупок для обеспечения государственных и муниципальных нужд» // Информационно-справочная система «Консультант Плюс».

3. Агеева Е. В. Цифровизация финансово-кредитной сферы в современной России. Монография / Агеева Е. В. , Афанасова М. А. , Баландина А. С. , Балашова Н. В. , Баннова К. А. — М.: Директ-Медиа, 2019. - 407с.

4. Antonio Gomes Rodrigues, Bruno Demion (Milamber), Philippe Mouawad. Master Apache JMeter - From Load Testing to DevOps: Master performance testing with JMeter. Packt Publishing. 2019. - 468 p.

5. Meier J.D., Carlos Farre, Prashant Bansode, Scott Barber, and Dennis Rea. Performance Testing Guidance for Web Applications. Microsoft Press. 2007. - 288 p.

6. Philipp K. Janert. Data Analysis with Open Source Tools. O'Reilly Media. 2010. - 583 p.

7. The Art of Application Performance Testing. Molyneaux Ian. O'Reilly Media. 2009. - 158 p.

8. The Art of Application Performance Testing: From Strategy to Tools. Molyneaux Ian. O'Reilly Media. 2014. - 278 p.

9. Дадашева А.З., Финансы. Учебник. - М. : Вузовский учебник, НИЦ ИНФРА-М, 2016. - 393 с.

10. Дементьев Д.В., Бюджетная система Российской Федерации. Учебник. — М.: КНОРУС, 2017. - 332 с.

11. Курченко Л.Ф., Бюджетная система Российской Федерации. Субфедеральный и местный уровни: учебное пособие. — М.: Дашков и К, 2016. - 252 с.

12. Нешитой А.С., Бюджетная система Российской Федерации: учебник для бакалавров. — М.: Дашков и К, 2015. - 310 с.

13. Sarojadevi H. Performance Testing: Methodologies and Tools. Journal of information engineering and applications, 2011, 5 - pp. 5-13.

14. Мясников С. О., Намиот Д. Е. Инструменты нагрузочного тестирования // Прикладная информатика. — 2018. — Т. 13, № 1. — С. 92–102.

15. Пешкова Х.В., Бюджетное устройство государства как основа ведения публичного хозяйства // Финансовое право. - 2012. - №4. - С. 14-27.

16. Савчук И. В., Особенности тестирования производительности финансовых приложений, предназначенных для исполнения бюджета и управления бюджетным процессом в субъектах РФ и муниципальных образованиях // Молодой ученый. — 2020. — № 18 (308). — С. 24-25.

17. International Software Qualifications Board (2018). Сертифицированный тестировщик - Программа обучения Базового уровня. Тестирование производительности. Russian Software Testing Qualifications Board. Web: https://www.rstqb.org/ru/istqb-downloads.html

18. Automatic Workload Repository (AWR) in Oracle Database 10g [Электронный ресурс] Oracle-Base - Режим доступа: https://oracle-base.com/articles/10g/automatic-workload-repository-10g (дата обращения: 22.05.2020)

19. GATLING [Электронный ресурс] Gatling - Режим доступа: https://gatling.io/docs/current/ (дата обращения: 15.03.2020)

20. JMeter – How To Save Test Results To A Database [Электронный ресурс] Vinsguru - Режим доступа: https://www.testautomationguru.com/jmeter-save-results-to-a-database/ (дата обращения: 20.05.2020)

21. Oracle Enterprise Manager [Электронный ресурс] Patches IT Community - Режим доступа: https://oracle-patches.com/oracle/products/2941-oracle-enterprise-manager-2 (дата обращения: 22.05.2020)

22. Overload [Электронный ресурс] GitHub - Режим доступа: https://github.com/yandex/yandex-tank/wiki/Overload (дата обращения: 14.05.2020)

23. performance testing [Электронный ресурс] SearchSoftwareQuality - Режим доступа: https://searchsoftwarequality.techtarget.com/definition/performance-testing (дата обращения: 29.04.2020)

24. POWA — анализатор нагрузки СУБД PostgreSQL на основе pg_stat_statements [Электронный ресурс] nixp - Режим

доступа: https://www.nixp.ru/news/12712.html (дата обращения: 22.05.2020)

25. Tank [Электронный ресурс] Yandex Technologies - Режим доступа: https://tech.yandex.com/tank/ (дата обращения: 14.05.2020)

26. Tsung — цунами ручного приготовления [Электронный ресурс] Блог Ивана Увтуховичв. Заметки из жизни DevOps-консультанта - Режим доступа: http://evtuhovich.ru/blog/2014/04/02/tsung (дата обращения: 06.05.2020)

27. Types of Performance Testing [Электронный ресурс] Software Testing Blog by Cigniti Technologies - Blog on Software Testing industry by leading Independent Software Testing Company Cigniti Technologies. - Режим доступа: https://www.cigniti.com/blog/types-of-performance-testing/ (дата обращения: 30.04.2020)

28. What is Grafana? [Электронный ресурс] GrafanaLabs - Режим доступа: https://grafana.com/docs/grafana/latest/getting-started/what-is-grafana/ (дата обращения: 19.05.2020)

29. Zabbix — мощный инструмент для мониторинга ИТ-инфраструктуры [Электронный ресурс] SysAdminTips.ru - Советы и подсказки системного администратора - Режим доступа: https://sysadmintips.ru/zabbix-instrument-dlya-monitoringa-it-infrastruktury.html (дата обращения: 18.05.2020)

30. Введение в исследование производительности [Электронный ресурс] ХАБР - Режим доступа:

https://habr.com/ru/company/yamoney/blog/433436/ (дата обращения: 20.04.2020)

31. Использование Apache Bench для тестирования нагрузки на веб-сервер [Электронный ресурс] Блог университета SEDICOMM - Режим доступа: https://www.performance-lab.ru/blog/load-testing/testirovanie-proizvoditelnosti (дата обращения: 08.05.2020)

32. Нагрузочное тестирование с Yandex.Tank и JMeter [Электронный ресурс] GitHub Gist - Режим доступа: https://gist.github.com/sameoldmadness/9abeef4c2125bc760ba2f09ee1150330 (дата обращения: 21.05.2020)

33. Нагрузочное тестирование vs Тестирование производительности [Электронный ресурс] PerfomanceLab - Режим доступа: https://bftcom.com/products/upravlenie-gosudarstvennymi-finansami/byudzhetnoe-planirovanie/?sphrase_id=6394 (дата обращения: 23.04.2020)

34. Система мониторинга Zabbix для начинающих [Электронный ресурс] Блог компании EternalHost - Режим доступа: https://eternalhost.net/blog/sistemnoe-administrirovanie/zabbix-chto-eto (дата обращения: 15.05.2020)

35.

Приложения

Приложение 1

Бизнес-сценарии для нагрузочного тестирования

Сценарий 1. Создание бюджетной заявки (БЗ)	1. Запуск клиента «АЦК-Планирование» и авторизация; В меню сверху раскрыть выпадающий список "Расходы"; 2. В списке "Расходы" открыть пункт меню "Бюджетные заявки"; 3. В открывшейся форме нажать на кнопку создания нового документа "Новый"; 4. В открывшейся форме в поле "Бланк расходов" вызвать "Справочник бланков расходов"; 5. В дереве бланков расходов справочника выбрать бланк расходов с типом "Смета" у которого есть вышестоящий бланк и его тип "Роспись" и нажать на кнопку ОК; 6. В форме создания новой БЗ нажать на кнопку создания новой строки документа "Новый"; 7. В открывшейся форме в поле "Наименование полномочия, расходного обязательства" вызвать справочник "Расходные обязательства"; 8. Выбрать из списка доступное расходное обязательство и нажать на кнопку "Выбрать"; 9. В форме создания новой строки документа перейти на поле "Сумма (1-ый год планирования)" и внести в него положительную сумму; 10. В форме создания новой строки документа нажать кнопку "ОК"; 11. В форме создания новой бюджетной заявки нажать кнопку "Применить"; 12. Выход из системы.
Сценарий 2. Обработка БЗ	1. Запуск клиента «АЦК-Планирование» и авторизация ; 2. В меню сверху раскрыть выпадающий список "Расходы"; 3. В списке "Расходы" открыть пункт меню "Бюджетные заявки"; 4. В открывшейся форме нажать на кнопку создания нового документа "Новый"; 5. В открывшейся форме в поле "Бланк расходов" вызвать "Справочник бланков расходов"; 6. В дереве бланков расходов справочника выбрать

	бланк расходов с типом "Смета" у которого есть вышестоящий бланк и его тип "Роспись" и нажать на кнопку ОК; 7. В форме создания новой БЗ нажать на кнопку создания новой строки документа "Новый"; 8. В открывшейся форме в поле "Наименование полномочия, расходного обязательства" вызвать справочник "Расходные обязательства"; 9. Выбрать из списка доступное расходное обязательство и нажать на кнопку "Выбрать"; 10. В форме создания новой строки документа перейти на поле "Сумма (1-ый год планирования)" и внести в него положительную сумму; 11. В форме создания новой строки документа нажать кнопку "ОК"; 12. В форме создания новой бюджетной заявки нажать кнопку "Применить"; 13. В форме создания новой бюджетной заявки нажать кнопку "Действие" и выбрать действие "Обработать"; 14. В форме создания новой бюджетной заявки нажать кнопку "Действие" и выбрать действие "Направить на согласование"; 15. В форме создания новой бюджетной заявки нажать кнопку "ОК"; 16. Выход из системы.
Сценарий 3. Создание сводной бюджетной заявки (СБЗ)	1. Запуск клиента «АЦК-Планирование» и авторизация; 2. В меню сверху раскрыть выпадающий список "Расходы"; 3. В списке "Расходы" открыть пункт меню "Бюджетные заявки"; 4. В открывшейся форме нажать на кнопку создания нового документа "Новый"; 5. В открывшейся форме в поле "Бланк расходов" вызвать "Справочник бланков расходов"; 6. В дереве бланков расходов справочника выбрать бланк расходов с типом "Смета" у которого есть вышестоящий бланк и его тип "Роспись" и нажать на кнопку ОК; 7. В форме создания новой БЗ нажать на кнопку создания новой строки документа "Новый"; 8. В открывшейся форме в поле "Наименование полномочия, расходного обязательства" вызвать справочник "Расходные обязательства"; 9. Выбрать из списка доступное расходное обязательство и нажать на кнопку "Выбрать"; 10. В форме создания новой строки документа перейти на поле "Сумма (1-ый год планирования)" и внести в

	него положительную сумму; 11. В форме создания новой строки документа нажать кнопку "ОК"; 12. В форме создания новой бюджетной заявки нажать кнопку "Применить"; 13. В форме создания новой бюджетной заявки нажать кнопку "Действие" и выбрать действие "Обработать"; 14. В форме создания новой бюджетной заявки нажать кнопку "Действие" и выбрать действие "Направить на согласование"; 15. В форме создания новой бюджетной заявки нажать кнопку "ОК"; 16. В списке "Расходы" открыть пункт меню "Распорядитель" - "Формирование сводных бюджетных заявок"; 17. В открывшемся АРМ-е, в блоке формирования в поле "Бланк расходов" выбрать бланк расходов с типом "Роспись", являющийся вышестоящим для созданной бюджетной заявки; 18. В гриде со списком документов выбрать созданный документ и нажать на кнопку "Сформировать"; 19. Выход из системы.
Сценарий 4. Обработка СБЗ	1. Запуск клиента «АЦК-Планирование» и авторизация ; 2. В меню сверху раскрыть выпадающий список "Расходы"; 3. В списке "Расходы" открыть пункт меню "Бюджетные заявки"; 4. В открывшейся форме нажать на кнопку создания нового документа "Новый"; 5. В открывшейся форме в поле "Бланк расходов" вызвать "Справочник бланков расходов"; 6. В дереве бланков расходов справочника выбрать бланк расходов с типом "Смета" у которого есть вышестоящий бланк и его тип "Роспись" и нажать на кнопку ОК; 7. В форме создания новой БЗ нажать на кнопку создания новой строки документа "Новый"; 8. В открывшейся форме в поле "Наименование полномочия, расходного обязательства" вызвать справочник "Расходные обязательства"; 9. Выбрать из списка доступное расходное обязательство и нажать на кнопку "Выбрать"; 10. В форме создания новой строки документа перейти на поле "Сумма (1-ый год планирования)" и внести в него положительную сумму; 11. В форме создания новой строки документа нажать кнопку "ОК";

	12. В форме создания новой бюджетной заявки нажать кнопку "Применить"; 13. В форме создания новой бюджетной заявки нажать кнопку "Действие" и выбрать действие "Обработать"; 14. В форме создания новой бюджетной заявки нажать кнопку "Действие" и выбрать действие "Направить на согласование"; 15. В форме создания новой бюджетной заявки нажать кнопку "ОК"; 16. В списке "Расходы" открыть пункт меню "Распорядитель" - "Формирование сводных бюджетных заявок"; 17. В открывшемся АРМ-е, в блоке формирования в поле "Бланк расходов" выбрать бланк расходов с типом "Роспись", являющийся вышестоящим для созданной бюджетной заявки; 18. В гриде со списком документов выбрать созданный документ и нажать на кнопку "Сформировать"; 19. В открывшемся окне редактирования сводной бюджетной заявки нажать кнопку "Действие" и выбрать действие "Направить на согласование"; 20. В форме редактирования сводной бюджетной заявки нажать кнопку "Действие" и выбрать действие "Завершить обработку"; 21. В форме редактирования сводной бюджетной заявки нажать кнопку "Действие" и выбрать действие "Утвердить"; 22. В открывшемся окне подтверждения утверждения документа нажимаем кнопку "ОК"; 23. В открывшейся форме утверждения документа в поле "Дата" выбираем дату равную дате документа Сводная бюджетная заявка; 24. Выход из системы.
Сценарий 5. Создание двух бюджетных заявок на изменение ассигнований (БЗнАИ)	1. Запуск клиента «АЦК-Планирование» и авторизация ; 2. В списке "Расходы" открыть пункт меню "Бюджетные заявки на изменение ассигнований"; 3. В открывшейся форме нажать на кнопку создания нового документа "Новый"; 4. В открывшейся форме в поле "Бланк расходов" вызвать "Справочник бланков расходов" В дереве справочника бланков расходов, выбрать бланк расходов с типом "Смета" аналогичный выбранному в БЗ; 5. В поле "Версия" вызвать справочник "Версии планирования расходов"; 6. Выбрать версию, имеющую признак "Актуальная", нажать кнопку "Выбрать";

	7. В форме создания новой БЗнИА нажать на кнопку создания новой строки документа "Новый"; 8. В открывшейся форме в поле "Наименование полномочия, расходного обязательства" вызвать справочник "Расходные обязательства"; 9. Выбрать из списка расходное обязательство аналогичное выбранному в БЗ; 10. Перевести курсор на поле "Проект изменений на очередной плановый/текущий период" и вызвать форму "Проект изменений на очередной плановый/текущий период"; 11. В форме создания новой строки документа перейти на поле "Сумма (1-ый год планирования)" и внести в него отрицательную сумму, по модулю равную сумме, указанной в БЗ; 12. В форме создания новой строки документа нажать кнопку "ОК"; 13. В форме создания новой бюджетной заявки на изменение ассигнований нажать кнопку "Применить"; 14. В форме списка БЗнИА нажать на кнопку создания нового документа "Новый"; 15. В открывшейся форме в поле "Бланк расходов" вызвать "Справочник бланков расходов"; 16. В дереве бланков расходов справочника бланков расходов выбрать бланк расходов с типом "Смета" аналогичный выбранному в БЗ; 17. В поле "Версия" вызвать справочник "Версии планирования расходов"; 18. Выбрать версию, имеющую признак "Актуальная", нажать кнопку "Выбрать"; 19. В форме создания новой БЗнИА нажать на кнопку создания новой строки документа "Новый"; 20. В открывшейся форме в поле "Наименование полномочия, расходного обязательства" вызвать справочник "Расходные обязательства"; 21. Выбрать из списка доступное расходное обязательство отличное от выбранного в БЗ; 22. В поле КФСР вызвать справочник "Функциональная классификация расходов" выбрать из списка доступный код нижнего уровня; 23. Перевести курсор на поле "Проект изменений на очередной плановый/текущий период" и вызвать форму "Проект изменений на очередной плановый/текущей период"; 24. В форме создания новой строки документа перейти на поле "Сумма (1-ый год планирования)" и внести в него положительную сумму равную сумме,

	указанной в БЗ; 25. В форме создания новой строки документа нажать кнопку "ОК"; 26. В форме создания новой бюджетной заявки нажать кнопку "Применить"; 27. Выход из системы.
Сценарий 6. Обработка двух БЗнИА	1. Запуск клиента «АЦК-Планирование» и авторизация ; 2. В списке "Расходы" открыть пункт меню "Бюджетные заявки на изменение ассигнований" 3. В открывшейся форме нажать на кнопку создания нового документа "Новый"; 4. В открывшейся форме в поле "Бланк расходов" вызвать "Справочник бланков расходов"; 5. В дереве справочника бланков расходов, выбрать бланк расходов с типом "Смета" аналогичный выбранному в БЗ; 6. В поле "Версия" вызвать справочник "Версии планирования расходов"; 7. Выбрать версию, имеющую признак "Актуальная", нажать кнопку "Выбрать"; 8. В форме создания новой БЗнИА нажать на кнопку создания новой строки документа "Новый"; 9. В открывшейся форме в поле "Наименование полномочия, расходного обязательства" вызвать справочник "Расходные обязательства"; 10. Выбрать из списка расходное обязательство аналогичное выбранному в БЗ; 11. Перевести курсор на поле "Проект изменений на очередной плановый/текущий период" и вызвать форму "Проект изменений на очередной плановый/текущий период"; 12. В форме создания новой строки документа перейти на поле "Сумма (1-ый год планирования)" и внести в него отрицательную сумму, по модулю равную сумме, указанной в БЗ; 13. В форме создания новой строки документа нажать кнопку "ОК"; 14. В форме создания новой бюджетной заявки на изменение ассигнований нажать кнопку "Применить"; 15. В форме создания новой бюджетной заявки на изменение ассигнование нажать кнопку "Действие" и выбрать действие "Обработать"; 16. В форме создания новой бюджетной заявки на изменение ассигнований нажать кнопку "Действие" и выбрать действие "Направить на согласование"; 17. В форме создания новой бюджетной заявки на

	изменение ассигнований нажать кнопку "ОК"; 18. В форме списка БЗнИА нажать на кнопку создания нового документа "Новый"; 19. В открывшейся форме в поле "Бланк расходов" вызвать "Справочник бланков расходов" 20. В дереве бланков расходов справочника бланков расходов выбрать бланк расходов с типом "Смета" аналогичный выбранному в БЗ; 21. В поле "Версия" вызвать справочник "Версии планирования расходов"; 22. Выбрать версию, имеющую признак "Актуальная", нажать кнопку "Выбрать"; 23. В форме создания новой БЗнИА нажать на кнопку создания новой строки документа "Новый"; 24. В открывшейся форме в поле "Наименование полномочия, расходного обязательства" вызвать справочник "Расходные обязательства"; 25. Выбрать из списка доступное расходное обязательство отличное от выбранного в БЗ; 26. В поле КФСР вызвать справочник "Функциональная классификация расходов" выбрать из списка доступный код нижнего уровня; 27. Перевести курсор на поле "Проект изменений на очередной плановый/текущий период" и вызвать форму "Проект изменений на очередной плановый/ткущий период"; 28. В форме создания новой строки документа перейти на поле "Сумма (1-ый год планирования)" и внести в него положительную сумму равную сумме, указанной в БЗ; 29. В форме создания новой строки документа нажать кнопку "ОК"; 30. В форме создания новой бюджетной заявки нажать кнопку "Применить"; 31. В форме создания новой бюджетной заявки нажать кнопку "Действие" и выбрать действие "Обработать"; 32. В форме создания новой бюджетной заявки нажать кнопку "Действие" и выбрать действие "Направить на согласование"; 33. В форме создания новой бюджетной заявки нажать кнопку "ОК"; 34. Выход из системы.
Сценарий 7. Создание сводной бюджетной заявки на изменение ассигнований (СБЗнАИ)	1. Запуск клиента «АЦК-Планирование» и авторизация; 2. В списке "Расходы" открыть пункт меню "Бюджетные заявки на изменение ассигнований"; 3. В открывшейся форме нажать на кнопку создания нового документа "Новый";

	4. В открывшейся форме в поле "Бланк расходов" вызвать "Справочник бланков расходов"; 5. В дереве справочника бланков расходов, выбрать бланк расходов с типом "Смета" аналогичный выбранному в БЗ; 6. В поле "Версия" вызвать справочник "Версии планирования расходов"; 7. Выбрать версию, имеющую признак "Актуальная", нажать кнопку "Выбрать"; 8. В форме создания новой БЗнИА нажать на кнопку создания новой строки документа "Новый"; 9. В открывшейся форме в поле "Наименование полномочия, расходного обязательства" вызвать справочник "Расходные обязательства"; 10. Выбрать из списка расходное обязательство аналогичное выбранному в БЗ; 11. Перевести курсор на поле "Проект изменений на очередной плановый/текущий период" и вызвать форму "Проект изменений на очередной плановый/текущий период"; 12. В форме создания новой строки документа перейти на поле "Сумма (1-ый год планирования)" и внести в него отрицательную сумму, по модулю равную сумме, указанной в БЗ; 13. В форме создания новой строки документа нажать кнопку "ОК"; 14. В форме создания новой бюджетной заявки на изменение ассигнований нажать кнопку "Применить"; 15. В форме создания новой бюджетной заявки на изменение ассигнование нажать кнопку "Действие" и выбрать действие "Обработать"; 16. В форме создания новой бюджетной заявки на изменение ассигнований нажать кнопку "Действие" и выбрать действие "Направить на согласование"; 17. В форме создания новой бюджетной заявки на изменение ассигнований нажать кнопку "ОК"; 18. В списке "Расходы" открыть пункт меню "Распорядитель" - "Формирование сводных бюджетных заявок"; 19. В открывшемся АРМ-е в блоке формирования в поле "Бланк расходов" выбрать бланк расходов с типом "Роспись", являющийся вышестоящим для созданной бюджетной заявки; 20. В открывшемся АРМ в блоке формирования в поле "Тип документа" выбрать сводную бюджетную заявку на изменение ассигнований; 21. В гриде со списком документов выбрать созданный

	документ и нажать на кнопку "Сформировать"; 22. Выход из системы.
Сценарий 8. Обработка БЗнАИ	1. Запуск клиента «АЦК-Планирование» и авторизация; 2. В форме списка БЗнИА нажать на кнопку создания нового документа "Новый"; 3. В открывшейся форме в поле "Бланк расходов" вызвать "Справочник бланков расходов"; 4. В дереве бланков расходов справочника бланков расходов выбрать бланк расходов с типом "Смета" аналогичный выбранному в БЗ; 5. В поле "Версия" вызвать справочник "Версии планирования расходов"; 6. Выбрать версию, имеющую признак "Актуальная", нажать кнопку "Выбрать"; 7. В форме создания новой БЗнИА нажать на кнопку создания новой строки документа "Новый"; 8. В открывшейся форме в поле "Наименование полномочия, расходного обязательства" вызвать справочник "Расходные обязательства"; 9. Выбрать из списка доступное расходное обязательство отличное от выбранного в БЗ; 10. В поле КФСР вызвать справочник "Функциональная классификация расходов" выбрать из списка доступный код нижнего уровня; 11. Перевести курсор на поле "Проект изменений на очередной плановый/текущий период" и вызвать форму "Проект изменений на очередной плановый/текущей период"; 12. В форме создания новой строки документа перейти на поле "Сумма (1-ый год планирования)" и внести в него положительную сумму равную сумме, указанной в БЗ; 13. В форме создания новой строки документа нажать кнопку "ОК"; 14. В форме создания новой бюджетной заявки нажать кнопку "Применить"; 15. В форме создания новой бюджетной заявки нажать кнопку "Действие" и выбрать действие "Обработать"; 16. В форме создания новой бюджетной заявки нажать кнопку "Действие" и выбрать действие "Направить на согласование"; 17. В форме создания новой бюджетной заявки нажать кнопку "ОК"; 18. В списке "Расходы" открыть пункт меню "Распорядитель" - "Формирование сводных бюджетных заявок"; 19. В открывшемся АРМ в блоке формирования в поле

	"Бланк расходов" выбрать бланк расходов с типом "Роспись", являющийся вышестоящим для созданной бюджетной заявки; 20. В открывшемся АРМ в блоке формирования в поле "Тип документа" выбрать сводную бюджетную заявку на изменение ассигнований; 21. В гриде со списком документов выбрать созданный документ и нажать на кнопку "Сформировать"; 22. В открывшемся окне редактирования сводной бюджетной заявки нажать кнопку "Действие" и выбрать действие "Направить на согласование"; 23. В форме редактирования сводной бюджетной заявки нажать кнопку "Действие" и выбрать действие "Завершить обработку"; 24. В форме редактирования сводной бюджетной заявки нажать кнопку "Действие" и выбрать действие "Утвердить"; 25. В открывшемся окне подтверждения утверждения документа нажимаем кнопку "ОК"; 26. В открывшейся форме утверждения документа в поле "Дата" выбираем равную даты документа Сводная бюджетная заявка; 27. Выход из системы.
Сценарий 9. План Финансово-хозяйственной деятельности (ПФХД)	1. Запуск клиента «АЦК-Планирование» и авторизация; 2. Создание организации; 3. Создание Структуры ПФХД; 4. Создание ПФХД; 5. Обработка ПФХД; 6. Выход из системы.
Сценарий 10. Цепочка документов ГП/ПП/ОМ и БЗ из нее	1. Вход в меню «Главная – Планирование бюджета – Расходы – Формирование бюджета программно-целевым способом – ВЦП/АЦП/Основное мероприятие»; 2. Создание нового документа «Государственная (муниципальная) программа»; 3. Заполнение обязательных полей; 4. Из ЭД ГП (закладка "Перечень подпрограмм и основных мероприятия") создаётся ЭД Подпрограмма; 5. Из ЭД Подпрограмма (закладка "Перечень ВЦП/АЦП/Основных мероприятия") создаётся ЭД Основное Мероприятие (ОМ); 6. В ЭД ОМ на закладке «Мероприятия» формируется перечень Мероприятий, в разрезе которых формируются расходные строки; 7. На основании расходной строки формируется ЭД «Бюджетная заявка» (БЗ) по кнопке «Отобразить значения в бюджетных заявках» вкладки

	«Мероприятия» ЭД ОМ; 8. Обработка ЭД ГП до статуса «Утверждено»; 9. Выход из системы.
Сценарий 11. Государственное (муниципальное) задание (ГмЗ)	1. Запуск клиента «АЦК-Планирование» и авторизация; 2. Вход в меню «Главная – Планирование бюджета – Расходы – Бюджетные услуги – Государственное (муниципальное) задание» 3. Создание нового документа «Государственное (муниципальное) задание (ГмЗ)»; 4. Заполнение обязательных полей; 5. Перевод ЭД ГМЗ на статус «Утверждено»; 6. Выход из системы.

Приложение 2

XML структура скрипта нагрузочного теста, созданного с помощью Apache JMeter

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.0 r1840935">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test
Plan" enabled="true">
      <stringProp name="TestPlan.comments"></stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
      <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments" guiclass="ArgumentsPanel" testclass="Arguments"
testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments"/>
      </elementProp>
      <stringProp name="TestPlan.user_define_classpath"></stringProp>
    </TestPlan>
    <hashTree>
      <Arguments guiclass="ArgumentsPanel" testclass="Arguments"
testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments">
          <elementProp name="address" elementType="Argument">
            <stringProp name="Argument.name">address</stringProp>
            <stringProp name="Argument.value"></stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>


          <elementProp name="pass" elementType="Argument">
            <stringProp name="Argument.name">pass</stringProp>
            <stringProp name="Argument.value">v</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
            <stringProp name="Argument.desc">Пустой пароль </stringProp>
          </elementProp>
          <elementProp name="sysuserid" elementType="Argument">
            <stringProp name="Argument.name">sysuserid</stringProp>
            <stringProp name="Argument.value"></stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="actversion_id" elementType="Argument">
            <stringProp name="Argument.name">actversion_id</stringProp>
            <stringProp name="Argument.value"></stringProp>
            <stringProp name="Argument.desc">Актуальная версия
расходов</stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
          <elementProp name="variant_id" elementType="Argument">
            <stringProp name="Argument.name">variant_id</stringProp>
            <stringProp name="Argument.value"></stringProp>
            <stringProp name="Argument.desc">Вариант </stringProp>
            <stringProp name="Argument.metadata">=</stringProp>
          </elementProp>
      </Arguments>
      <hashTree/>
      <CSVDataSet guiclass="TestBeanGUI" testclass="CSVDataSet"
testname="pldirection.csv" enabled="true">
```

```
        <stringProp name="filename"></stringProp>
        <stringProp name="fileEncoding">UTF-8</stringProp>
        <stringProp name="variableNames">pldirection_id</stringProp>
        <boolProp name="ignoreFirstLine">false</boolProp>
        <stringProp name="delimiter">;</stringProp>
        <boolProp name="quotedData">false</boolProp>
        <boolProp name="recycle">true</boolProp>
        <boolProp name="stopThread">false</boolProp>
        <stringProp name="shareMode">shareMode.all</stringProp>
      </CSVDataSet>
      <hashTree/>
      <ResultSaver guiclass="ResultSaverGui" testclass="ResultSaver"
testname="Сохранение ошибок сервера в файл" enabled="true">
        <stringProp name="FileSaver.filename">pl_fail_</stringProp>
        <boolProp name="FileSaver.errorsonly">true</boolProp>
        <boolProp name="FileSaver.skipautonumber">false</boolProp>
        <boolProp name="FileSaver.skipsuffix">false</boolProp>
        <boolProp name="FileSaver.successonly">false</boolProp>
        <boolProp name="FileSaver.addTimstamp">true</boolProp>
      </ResultSaver>
      <hashTree/>
      <ResponseAssertion guiclass="AssertionGui"
testclass="ResponseAssertion" testname="error" enabled="true">
        <collectionProp name="Asserion.test_strings">
          <stringProp name="-1294635214">error:</stringProp>
        </collectionProp>
        <stringProp
name="Assertion.test_field">Assertion.response_data</stringProp>
        <boolProp name="Assertion.assume_success">false</boolProp>
        <intProp name="Assertion.test_type">6</intProp>
        <stringProp name="Assertion.custom_message"></stringProp>
      </ResponseAssertion>
      <hashTree/>
      <kg.apc.jmeter.threads.SteppingThreadGroup
guiclass="kg.apc.jmeter.threads.SteppingThreadGroupGui"
testclass="kg.apc.jmeter.threads.SteppingThreadGroup" testname="STRESS"
enabled="true">
        <stringProp
name="ThreadGroup.on_sample_error">startnextloop</stringProp>
        <stringProp name="ThreadGroup.num_threads">250</stringProp>
        <stringProp name="Threads initial delay">0</stringProp>
        <stringProp name="Start users count">5</stringProp>
        <stringProp name="Start users count burst">0</stringProp>
        <stringProp name="Start users period">4</stringProp>
        <stringProp name="Stop users count">250</stringProp>
        <stringProp name="Stop users period">1</stringProp>
        <stringProp name="flighttime">3600</stringProp>
        <stringProp name="rampUp">8</stringProp>
        <elementProp name="ThreadGroup.main_controller"
elementType="LoopController" guiclass="LoopControlPanel"
testclass="LoopController" testname="Loop Controller" enabled="true">
          <boolProp name="LoopController.continue_forever">false</boolProp>
          <intProp name="LoopController.loops">-1</intProp>
        </elementProp>
      </kg.apc.jmeter.threads.SteppingThreadGroup>
      <hashTree>
        <CounterConfig guiclass="CounterConfigGui" testclass="CounterConfig"
testname="Counter" enabled="true">
          <stringProp name="CounterConfig.start">0</stringProp>
          <stringProp name="CounterConfig.end">4999</stringProp>
          <stringProp name="CounterConfig.incr">1</stringProp>
```

```
          <stringProp name="CounterConfig.name">login</stringProp>
          <stringProp name="CounterConfig.format">0</stringProp>
          <boolProp name="CounterConfig.per_user">false</boolProp>
        </CounterConfig>
        <hashTree/>
        <UniformRandomTimer guiclass="UniformRandomTimerGui"
testclass="UniformRandomTimer" testname="Uniform Random Timer"
enabled="true">
          <stringProp name="ConstantTimer.delay">500</stringProp>
          <stringProp name="RandomTimer.range">500</stringProp>
        </UniformRandomTimer>
        <hashTree/>
        <ThroughputController guiclass="ThroughputControllerGui"
testclass="ThroughputController" testname="Минфин РК" enabled="true">
          <intProp name="ThroughputController.style">1</intProp>
          <boolProp name="ThroughputController.perThread">false</boolProp>
          <intProp name="ThroughputController.maxThroughput">1</intProp>
          <FloatProperty>
            <name>ThroughputController.percentThroughput</name>
            <value>36.03</value>
            <savedValue>0.0</savedValue>
          </FloatProperty>
        </ThroughputController>
        <hashTree>
          <GenericController guiclass="LogicControllerGui"
testclass="GenericController" testname="Сценарий 1" enabled="true"/>
          <hashTree>
            <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="check_sessions_count - nobody"
enabled="true">
              <boolProp name="HTTPSampler.postBodyRaw">true</boolProp>
              <elementProp name="HTTPsampler.Arguments"
elementType="Arguments">
                <collectionProp name="Arguments.arguments">
                  <elementProp name="" elementType="HTTPArgument">
                    <boolProp
name="HTTPArgument.always_encode">false</boolProp>
                    <stringProp name="Argument.value"></stringProp>
                    <stringProp name="Argument.metadata">=</stringProp>
                  </elementProp>
                </collectionProp>
              </elementProp>
              <stringProp name="HTTPSampler.domain">${address}</stringProp>
              <stringProp name="HTTPSampler.port">${port}</stringProp>
              <stringProp name="HTTPSampler.protocol">http</stringProp>
              <stringProp name="HTTPSampler.contentEncoding"></stringProp>
              <stringProp name="HTTPSampler.path">/exec</stringProp>
              <stringProp name="HTTPSampler.method">POST</stringProp>
              <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
              <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
              <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
              <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
              <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
              <stringProp name="HTTPSampler.connect_timeout"></stringProp>
              <stringProp name="HTTPSampler.response_timeout"></stringProp>
            </HTTPSamplerProxy>
            <hashTree>
              <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager"
testname="HTTP Header Manager" enabled="true">
                <collectionProp name="HeaderManager.headers">
                  <elementProp name="method" elementType="Header">
```

```
                    <stringProp name="Header.name">method</stringProp>
                    <stringProp name="Header.value">job_process</stringProp>
                  </elementProp>
                  <elementProp name="Content-Type" elementType="Header">
                    <stringProp name="Header.name">Content-Type</stringProp>
                    <stringProp name="Header.value">application/octet-
stream</stringProp>
                  </elementProp>
                  <elementProp name="sessionid" elementType="Header">
                    <stringProp name="Header.name">sessionid</stringProp>
                    <stringProp name="Header.value"></stringProp>
                  </elementProp>
                  <elementProp name="Accept-Encoding" elementType="Header">
                    <stringProp
name="Header.name">Accept-Encoding</stringProp>
                    <stringProp name="Header.value">identity</stringProp>
                  </elementProp>
                  <elementProp name="Accept" elementType="Header">
                    <stringProp name="Header.name">Accept</stringProp>
                    <stringProp name="Header.value">*/*</stringProp>
                  </elementProp>
                  <elementProp name="User-Agent" elementType="Header">
                    <stringProp name="Header.name">User-Agent</stringProp>
                    <stringProp name="Header.value">Mozilla/3.0 (compatible;
Indy Library)</stringProp>
                  </elementProp>
                </collectionProp>
              </HeaderManager>
            </hashTree>
          </hashTree>
        </hashTree>
      </hashTree>
    </hashTree>
  </hashTree>
</jmeterTestPlan>
```

Приложение 3

Справка о публикации научной статьи

ООО «ИЗДАТЕЛЬСТВО МОЛОДОЙ УЧЕНЫЙ»

ИНН/КПП: 7536104558/166001001
420029, г. Казань, ул. Акад. Кирпичникова, 25
Тел./факс: (843) 500-57-53, 8-800-555-14-87
E-mail: info@moluch.ru
Сайт: https://moluch.ru/
Исх. № 69405 от 19.05.2020

СПРАВКА

Подтверждаем, что статья **«Особенности тестирования производительности финансовых приложений, предназначенных для исполнения бюджета и управления бюджетным процессом в субъектах РФ и муниципальных образованиях»** (автор: Савчук Иван Валерьевич) опубликована в **международном научном журнале «Молодой ученый» №18 (308), май 2020 г.** (стр. 24-25) , URL: https://moluch.ru/archive/308/69405/ . Журнал «Молодой ученый» выходит в печатном виде (ISSN 2072-0297, свидетельство о регистрации СМИ ПИ № ФС77-38059 от 11 ноября 2009 г., выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций). Журнал размещается на портале elibrary.ru, на данный момент не входит в РИНЦ.

Главный редактор: /к.т.н. Ахметов И.Г./

Исп.: Шульга О.А.