DOI: https://doi.org/10.17816/2074-0530-109890

Оригинальное исследование

К вопросу повышения эффективности и безопасности эксплуатации тоннельных эскалаторов метрополитена. Архитектура программного обеспечения

В.А. Попов, В.В. Еланцев

Петербургский государственный университет путей сообщения Императора Александра I, Санкт-Петербург, Российская Федерация

АННОТАЦИЯ

Обоснование. Цифровая трансформация бизнес-процессов предприятий всех форм собственности — неотъемлемое условие их существования при стремительных внешних и внутренних преобразованиях. Не исключение и Петербургский метрополитен, имеющий в своем составе стареющий парк эскалаторов, которому внедрение цифровых технологий, сопровождаемое оптимизацией системы управления основными технологическими процессами, может обеспечить баланс между всевозрастающими затратами и ограниченными, не всегда ритмично поступающими, ресурсами.

Цель работы — установить один из возможных вариантов архитектуры программного обеспечения для планирования и контроля проведения технического обслуживания и ремонта эскалаторного хозяйства метрополитена. Вместе с тем объектом исследования в работе выступает эскалаторное хозяйство, а предметом — автоматизация системы технического обслуживания и ремонта.

Материалы и методы. В качестве методов в работе выступают анализ современных архитектурных решений, используемых при проектировании программного обеспечения (материалов), и синтез на их основе микросервисной архитектуры, реализованной инструментарием из области машинного обучения и искусственного интеллекта.

Результаты. Значимым для практического применения результатом работы является предложенная схема двухконтурного ИТ-ландшафта информационной системы и структурная схема программного обеспечения, реализующего внутренний обрабатывающий контур. Предлагаемый вариант программного обеспечения предназначен для регулярного использования в качестве инструмента мониторинга и контроля реестра объектов управления (работ), создающих и не создающих ценность для компании (обеспечивающих/не обеспечивающих транспортировку пассажиропотоков). При этом реализация предложенного схемотехнического решения может осуществляться с использованием готовых и свободно распространяемых компонентов, значительно сокращающих время проектирования и стоимость его создания и эксплуатации.

Заключение. Полученные результаты могут являться одним из вариантов воплощения концепции цифровой трансформации системы технического обслуживания и ремонта эскалаторного хозяйства метрополитена.

Ключевые слова: *эскалатор; микросервисная архитектура; конвейер данных; добыча данных; обогащение данных; BI-инструменты.*

Как цитировать:
Попов В.А., Еланцев В.В. К вопросу повышения эффективности и безопасности эксплуатации тоннельных эскалаторов метрополитена. Архитектура программного обеспечения // Известия МГТУ «МАМИ». 2023. Т. 17, № 1. С. 63–76. DOI: https://doi.org/10.17816/2074-0530-109890

Рукопись получена: 15.08.2022 **Рукопись одобрена:** 04.09.2022 **Опубликована онлайн:** 18.09.2022

DOI: https://doi.org/10.17816/2074-0530-109890

Original study article

Increasing efficiency and safety of operation of underground tunnel escalators. The software architecture

Valery A. Popov, Valentin V. Elantsev

Emperor Alexander I St. Petersburg State Transport University, Saint Petersburg, Russian Federation

ABSTRACT

BACKGROUND: An essential condition of existence for enterprises of all forms in the face of rapid external and internal transformations is digital reformation of their business processes. The Saint Petersburg subway is no exception, which has an aging fleet of escalators, for which the introduction of digital technologies, accompanied by the optimization of the control system for the main technological processes, can provide a balance between ever-increasing costs and limited, not always rhythmically flowing supplies.

AIMS: To establish one of the possible options for the software architecture for planning and monitoring maintenance and repair of the subway escalator fleet. At the same time, the object of research in the study is the subway escalator fleet, and the subject is automation of the maintenance and repair system.

METHODS: Analysis of modern architectural solutions used in the design of software (materials), and the synthesis on their basis of a microservice architecture implemented by tools taken from the field of machine learning and artificial intelligence are the methods used in this study.

RESULTS: A significant result of the work for practical application is the proposed scheme of the two-circuit IT landscape of the information system and the block diagram of the software implementing the internal processing circuit. The proposed version of the software is intended for regular use as a tool for monitoring and controlling the register of management objects (works) that create and do not create value for the company (providing/not providing transportation of passenger traffic). At the same time, the implementation of the proposed circuit design solution can be carried out using ready-to-use and free software components, significantly reducing the design time and the cost of its creation and operation.

CONCLUSIONS: The results obtained may be one of the options for implementing the concept of digital transformation of the maintenance and repair system of the subway escalator fleet.

Keywords: *escalator, microservice architecture, data pipeline, data mining, data enrichment, BI-tools.*

To cite this article:
Popov VA, Elantsev VV. Increasing efficiency and safety of operation of underground tunnel escalators. The software architecture. *Izvestiya MGTU «MAMI»*. 2023;17(1): 63–76. DOI: https://doi.org/10.17816/2074-0530-109890

Received: 15.08.2022 **Accepted:** 04.09.2022 **Published online:** 18.09.2022

ВВЕДЕНИЕ

Стремительно меняющиеся внешние и внутренние экономические реалии нарушают устоявшиеся логистические цепочки, оказывая тем самым существенное негативное влияние на экономику не только коммерческих, но и государственных предприятий, включая городской пассажирский транспорт [1, 2].

В этих условиях для обеспечения своего существования и поддержания надлежащего уровня качества результатов своей деятельности на первый план для организаций выходят задачи динамичной перестройки внутренних бизнес-процессов [3].

Основным направлением при перестройке бизнес-процессов является их картирование [4] для установления узких мест при использовании собственных ресурсов, и последующая цифровизация системы поддержки принятия решений на базе отечественного программного обеспечения, способствующего повышению эффективности использования циркулирующих информационных потоков для эффективной утилизации имеющихся ресурсов.

При этом качество и эффективность подготавливаемых программным обеспечением рекомендаций напрямую зависит от архитектуры информационной системы [5].

Архитектура [6] в свою очередь определяет интенсивность использования вычислительных ресурсов для обслуживания пользователей, производительность, отказоустойчивость и масштабируемость инфраструктуры, а также скорость выпуска обновлений (релизов). Выбор архитектуры также влияет на величину потребных ресурсов на разработку системы, ее обслуживание и развитие, включая размер наиболее эффективных команд их структуру и формат взаимоотношений между ними.

Цель данной статьи — установить один из возможных вариантов архитектуры программного обеспечения для планирования и контроля проведения технического обслуживания и ремонта эскалаторного хозяйства метрополитена.

Объектом исследования является эскалаторное хозяйство метрополитена. В свою очередь предметом исследования является автоматизация системы технического обслуживания и ремонта.

1. Краткий обзор существующих архитектур в приложении к цели исследования

1.1 Теоретические основы

В основе построения любой архитектуры сложного многофункционального программного обеспечения лежит абстрагирование, а также дробление и инкапсуляция (изоляция), иными словами, расчленение сложных действий на более простые (ограничение бизнес-контекста [7]), имеющие однозначно определяемые входы/выходы и не влияющие на другие, управляемые с более высокого уровня (функциями нижнего уровня описываются сложные действия верхнего (абстрактного) уровня).

Несоблюдение изолированности отдельных логических единиц, разграничивающих (декомпозирующих) логику и ответственность разработчиков, приводит к непредсказуемому влиянию изменений на поведение всей системы и появлению ошибок (сбоев) [8].

Так, к примеру, на рис. 1 выделены уровни абстракции [9], обеспечивающие управляемость и снижение сложности программного обеспечения за счет вынесения основной логики на верхние слои и распределения рутинной и малозначимой логики на нижних. На верхнем уровне исполняется бизнес-логика (полезные функции — ценность), нижние уровни реализуют вспомогательные функции и взаимодействие с инфраструктурными системами (база данных, хранилище и т. д.). Программные единицы верхнего уровня изолированы от исполнения нижестоящих, которые являются для него черным ящиком, обеспечивающим использование определенного набора функций, уменьшая общую связность между программными единицами. При этом элементы одного уровня абстракции взаимодействуют между собой только через вышестоящий (управляющий) уровень, эти особенности способствуют удобству для поддержки, развития и тестирования исходного кода программного обеспечения.

Разные архитектуры разделяют логику на компоненты и слои в зависимости от (используемого технологического стека [10]), среды и жизненного цикла программного обеспечения.

В приложение к цели исследования особенности представленного выше разделения логики рассмотрим на примере клиент-серверного подхода.

При клиент-серверном подходе клиентские и серверные компоненты программного обеспечения функционируют физически раздельно, взаимодействуя

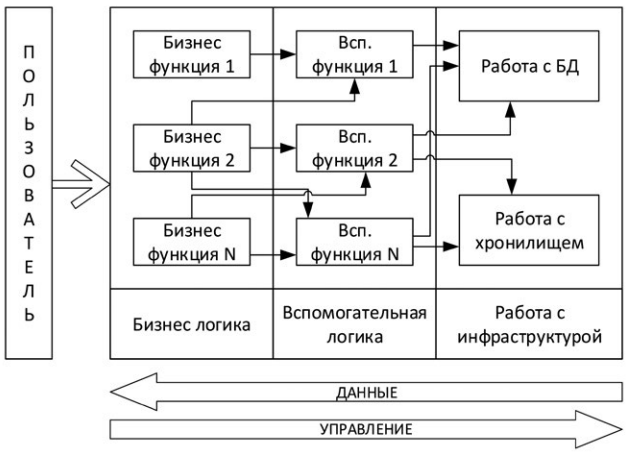

Рис. 1. Уровни абстракции.
Fig. 1. Abstraction levels.

DOI: https://doi.org/10.17816/2074-0530-109890

через согласованный набор команд — *API* (*Application Programming Interface*).

Данный подход характерен для программного обеспечения со сложной структурой и растущим трафиком использования, а также используется при наличии достаточных ресурсов и в качестве альтернативы монолитной архитектуры, реализуемой внутри одного модуля (программы).

Клиент-серверный подход [11], приведенный на рис. 2, разделяет логику работы на три подсистемы:

- серверная часть (*back-end — BE*) — основная логика (ценность);
- клиентская часть (*front-end — FE*) — пользовательский интерфейс (UX/UI дизайн);
- база данных (*BD*) — хранилище данных.

Таким образом, в одной подсистеме аккумулировано все, что касается взаимодействия с пользователем, а в другой — вся полезная логика (расчеты/преобразование данных и т. п.).

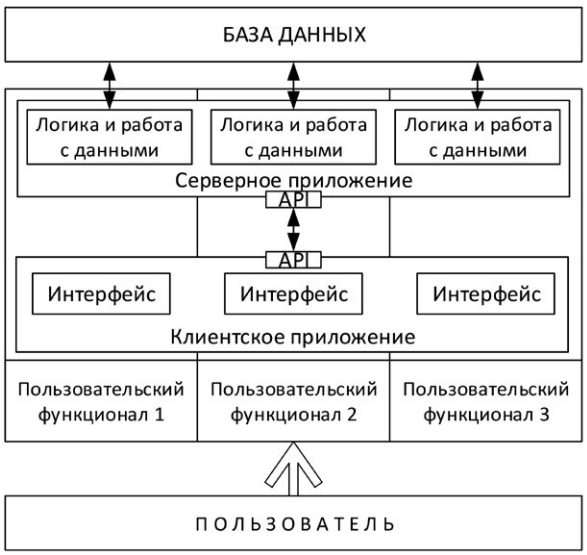

Рис. 2. Упрощенное представление разделения на FE и BE.
Fig. 2. A simplified image of division to FE and BE.

Клиент-серверный подход обеспечивает независимое выполнение работ при разработке/тестировании/поддержке в эксплуатации/рефакторинге разнопрофильными специалистами; повышение качества программного обеспечения за счет разграничения зон ответственности и вовлеченности; переход от веб-приложения к мобильному без дополнительных сложностей и ресурсов за счет изоляции *BE* от *FE* и использования унифицированных *API*; простоту адаптации под высокие нагрузки за счет изоляции бизнес логики в отдельные модули, которые проще горизонтально/вертикально масштабировать; уменьшение нагрузки на сетевую инфраструктуру за счет того, что исполняемая логика размещается на сервере.

К основным недостаткам клиент-серверного подхода относятся рост числа исполнителей для поддержки как *FE*, так и *BE* элементов, за счет чего увеличивается стоимость обслуживания, а также требуется использование дорогостоящего серверного оборудования.

Далее с учетом описанных выше особенностей клиент-серверного подхода в качестве базового паттерна разработки рассмотрим широко используемую сегодня, минимизирующую присущие предку недостатки микросервисную архитектуру программного обеспечения.

1.2 Микросервисная архитектура

Микросервисы позволяют оперативно реагировать на изменения бизнес-требований (ограниченного контекста), сохранять работоспособность, понятность, изолированность и инкапсулированность кода в отдельных связных модулях (сервисах) за счет соблюдения пределов ограниченных контекстов [12].

В базисе деления на сервисы лежит выделение основной ведущей системы (основная логика) и вспомогательных или выделение относительно равнозначные сервисов, отвечающих за свою функциональную область (рис. 3).

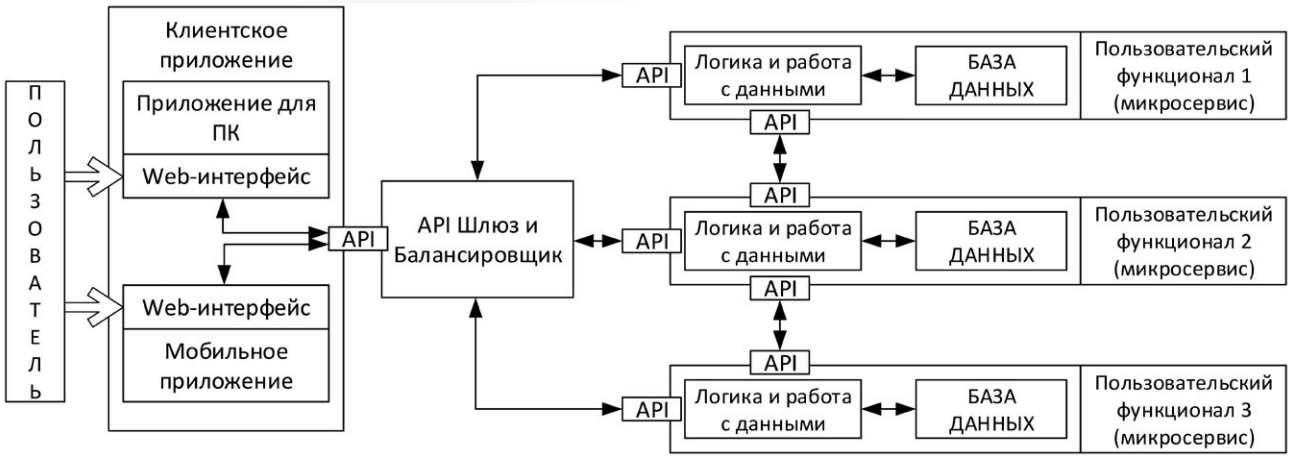

Рис. 3. Обобщенный пример микросервисной архитектуры.
Fig. 3. A general example of the microservice architecture.

В части размеров сервисов существует устоявшееся представление о том, что микросервис должен соответствовать ограниченному контексту, полностью понятному одном человеку. Иными словами, чем шире функциональность программного обеспечения, тем больше должно быть микросервисов в его составе. Вместе с тем, наиболее сложным при разработке микросервисов является именно определение их границ. Ограниченный контекст большой для понимания одним человеком необходимо членить на более мелкие, что соответствует концепциям архитектуры с эволюционным развитием и предметно-ориентированного программирования, подразумевающим постоянное изменение (рефакторинг/реинжиниринг) по мере углубления в предметную область и/или изменений бизнес-требований. При использовании архитектуры с эволюционным развитием изменения реализуются за счет объединения микросервисов при частом одновременном изменении нескольких микросервисов, разделение микросервисов при потребности разделения бизнес-логики для увеличения ценности и при наличии возможности, а также добавление через микросервис пилотного функционала, работающего определенное время.

Для микросервисной архитектуры также характерно разделение *FE* и *BE* на независимые свободно сопряженные приложения — сервисы/микросервисы, обслуживающие определенную функциональную область или отдельные функции, которые могут быть структурированы следующим образом:

1. Через распределение пользовательского интерфейса между несколькими микросервисами с сохранением взаимосвязи с *BE*. При такой структуре налаживается внутрипроцессное взаимодействие между *FE* и *BE*, но усложняется обеспечение связности пользовательского интерфейса (*UI*), и в случае перекрестных изменений границ *UI* приходится одновременно обновлять несколько микросервисов, создавая новые взаимосвязи и нарушая изолированность и независимость существующих микросервисов.
2. Через разделение пользовательского интерфейса между *FE* и *BE*, которое осуществляется таким образом, чтобы *UI* изолировался одним сервисом и взаимодействовал с *BE* через стандартные протоколы. При этом *UI* поддерживается целиком, не теряя связность. Для этой структуры взаимодействия между *FE* и *BE* осуществляется либо несколькими небольшими асинхронными *HTTP*-запросами, исключающими их блокировку, либо выделяется один большой запрос к специализированным сервисам (шлюз/агрегатор/кеш), собирающим данные со всей микросервисной экосистемы.

Вместе с тем описанные выше характеристики порождают особенности управления микросервисной архитектурой, представленные в табл. 1.

На основе описанных выше особенностей формируются следующие преимущества микросервисной архитектуры:

- **Частичное развертывание**, позволяющее обновлять программное обеспечение по частям.
- **Устойчивость при сбоях**, так как сбой одного микросервиса не приводит к сбою всего программного обеспечения.
- **Устойчивость инкапсуляции** ограниченного контекста за счет отсутствия общих состояний (*shared state*) между модулями и физического разделения между серверами.
- **Независимая эволюция подсистем** достигается за счет способности микросервисов раздельно развиваться и изменять обратную совместимость без поддержки старых версий всего программного обеспечения, так как старая версия отдельного сервиса может еще работать необходимое время.
- **Мультиплатформенность и гетерогенность**, предотвращающие возникновение тесных связей между микросервисами и позволяющие выбирать подходящие и удобные технические решения и экспериментировать с новыми технологиями, достигаются за счет использования различных способов взаимодействия, и языков программирования микросервисов, и их распределения по разным серверным узлам, а также корректного подбора инструментов мониторинга и хранения данных.
- **Независимая** друг от друга и от остального программного обеспечения **масштабируемость** микросервисов достигается за счет инкапсуляции определенных ограниченных контекстов в сервисы, работающие на разных узлах и взаимодействующие между собой по сети с помощью заранее согласованных и редко изменяемых *API*.
- **Эффективное использование облачных мощностей** за счет облачных контейнеров, обеспечивающих микросервисам обслуживание большой нагрузки и удобство масштабирования, позволяющих по запросу выделять необходимое количество вычислительной мощности для нужд микросервиса.

Из приведенного выше описания микросервисной архитектуры следует, что именно этот паттерн наилучшим образом походит для разработки, тестирования, разворачивания и эксплуатации многофункционального программного обеспечения для планирования и контроля проведения технического обслуживания эскалаторного хозяйства метрополитена.

DOI: https://doi.org/10.17816/2074-0530-109890

Таблица 1. Особенности управления
Table 1. Management features

Особенность	Описание
При управлении гибкостью технологий	С учетом использования разнообразного технического стека при разработке микросервисов из экосистемы при реализации одних и тех же задач должен использоваться одинаковый стек и технологические приемы.
При управлении нестабильностью интерфейса	При внесении изменений в *API* (ввод/вывод) необходимо контролировать влияние этого изменения на другие микросервисы и систему в целом.
При управлении согласованностью данных	В связи с тем, что большинство микросервисов обладает собственными базами данных, которые изолированы от других и доступны для внешнего взаимодействия (*CRU*D) только через соответствующие интерфейсы, общая степень устойчивости данных варьируется в зависимости от сервиса. Хранилища данных из серверных микросервисов могут частично или целиком копироваться в клиентские микросервисы. При изменении исходных хранилищ запускается обновление данных, скопированных клиентским микросервисом, посредством отправки сообщений и постановки их в очередь на прочтение. До тех пор, пока клиентский микросервис не получит и не прочтет сообщение, данные в нем будут устаревшими, а с исходным микросервисом не согласованными. Описанное выше постепенное выполнение изменений и несогласованность данных на протяжение короткого периода времени называется согласованием в конечном счете и имеет особое значение при разработке от серверной части до *UX*-уровней.
При управлении развертыванием и мониторингом	При реализации микросервисной архитектуры, обеспечивающей независимое обновление отдельного функционала программного обеспечения, особое значение имеет использование *DevOps*-культуры и технологий непрерывной интеграции/непрерывного развертывания (доставки) (*CI/CD*). Технологии, реализующие *CI/CD*, позволяют быстро вводить в эксплуатацию производственные мощности за счет оперативного мониторинга и перераспределения мощностей для разработки, тестирования, приемки и эксплуатации и быстро устанавливать программное обеспечение (обновления/релизы) за счет автоматизации сборки, упаковки, настройки специфических параметров окружения, конфигурируемых при развертывании. *DevOps*-культура подразумевает взаимодействие на стыке разработки, тестирования и эксплуатации. Культура формируется за счет стандартизации информационных окружений при разработке, тестировании и эксплуатации, а также объединения инструментов автоматизации для оперативного переноса программного обеспечения или отдельных сервисов через стадии жизненного цикла, обеспечивая быстрый выпуск версий (релизов).
При управлении организацией работ	Согласно перефразированному закону Конвея, микросервисисную архитектуру, порождает организационная структура, повторяющая ее. Иными словами, наиболее эффективным способом организации работы при создании и сопровождении программного обеспечения, построенного на основе микросервисной архитектуры, является выделение группы исполнителей на основе бизнес-возможностей для каждого или группы сервисов, имеющих возможность изменять (обновлять) отдельные микросервисы независимо от других (без дополнительных согласований и синхронизации) за счет межсервисного взаимодействия через *API*.

2. Архитектурное решение программного обеспечения для планирования и контроля проведения технического обслуживания и ремонта эскалаторного хозяйства метрополитена

Предлагаемая архитектура программного обеспечения является продолжением работы [13] и имеет два контура, включающих внешние информационные системы и внутренний обрабатывающий контур.

Внешний информационный контур является адаптацией типовых решений в области СЭД, ЭДО, *ERP* и прочих корпоративных сервисов и не представляет интерес для подробного рассмотрения в данной работе [14, 15].

Вместе с тем отметим, что во внешней информационной системе, на нулевом этапе в момент внедрения, на основе хранящихся в библиотеках шаблонах, на каждый эскалатор и другие объекты учета заводится отдельная учетная сущность (паспорт), в которой фиксируются базовые индивидуальные характеристики (анамнез) — глубина наклонного хода, тип эскалатора, количество ступеней, заводской номер, пассажиропоток и другая информация, включая информацию о рисках, и т. п.

Тогда же задается и «идеальная» модель технического состояния эскалаторного хозяйства, запечатленная в эталонных карточках шаблонов-трафаретов технических воздействия (субсущностях) из состава рекомендованного перечня работ, которая в свою очередь

определяет для каждой дистанции, станции, станционного выхода и группы однотипных эскалаторов «идеальную» модель технического состояния и рекомендации для его восстановления в случае отклонения какого-либо параметра или группы параметров от нормы.

Далее более подробно рассмотрим внутренний обрабатывающий контур, представленный программно-аппаратным комплексом.

На первом этапе, в процессе реальной эксплуатации, в рамках операционной и проектной деятельности, ответственными лицами из подразделений эскалаторного хозяйства, в соответствии с первичной/обновленной «идеальной» моделью технического состояния формируются перечни запрашиваемых работ — заполняются карточки шаблонов-трафаретов(наряды-допуски) на выполнение тех или иных работ. В результате внутри паспорта создаются множества отдельных субсущностей, соответствующие тому или иному виду работ, содержащие соответствующие им плановые и фактические значения параметров. При этом под операционной деятельностью понимается обслуживание, текущее содержание, дефектация, экспертизы и прочие работы, не требующие значительных затрат (включая капитальные). В данной работе под проектной деятельностью [16, 17, 18] понимаются ремонтные работы и реконструкции/модернизации, состоящие из совокупности скоординированных и управляемых видов деятельности с начальной и конечной датами, предпринятые для достижения соответствующей конкретным требованиям цели, включающие ограничения по срокам, стоимости и ресурсам и связанные со значительными затратами (включая капитальные). На этом этапе пользователь также видит динамику реализации и оценку выполненных работ в виде лингвистических характеристик, выставленных на основе количества баллов, отражающих объем работ, срок и ресурсы.

На втором этапе агрегированная потребность из состава запрашиваемого перечня работ, в виде сырых неподготовленных данных, через сервис управления доступом к данным попадает в сервис — конвейер данных, перемещаясь по которому, они размещаются на хранение и проходят ряд подготовительных процедур.

И сырые, и подготовленные данные с конвейера для первичной аналитической обработки (условные сита грохота/сепаратора), на третьем этапе, передаются в сервис управления мастер-данными, где раскладываются на составляющие элементы (фракционирование), верифицируются, оцениваются по уровню выполнения предыдущей потребности, соотносятся с эталонной моделью данных и дополняются служебной информацией. После чего «просеянные» данные обратно возвращаются на конвейер данных для формирования OLAP-кубов. Затем через сервис OLAP-кубы «просеянные» данные подаются в сервис добычи данных, внутри которого первично оценивается набор параметрической информации по принадлежности допусковым зонам и аномалиям поведения. При наличии данных об экспертизе промышленной безопасности (заполненных соответствующих субсущностях) и другой параметрической информации, полученной по результатам ранее выполненных работ, рассчитывается остаточный ресурс. После, с учетом ранее добытых данных и данных из хранилища, прогнозируется техническое состояние. Так как техническое состояние — это динамичная сущность, фиксируемая в электронных документах (сущностях/субсущностях) и определяемая в каждый момент времени комплексом плановых и фактических технических и экономических параметров, то задача сервиса — на основе предыдущей истории запросов и пользовательской активности формировать ранжированный перечень релевантных комплексов технических воздействий для поддержания заданного уровня, к примеру, на основе алгоритма Палех, Королев и *YATI*, используемых Яндексом, или их аналогов. При этом в зависимости от ранее принятых управленческих решений и уровня реализации, удовлетворения предыдущей потребности (утвержденных потребностей) алгоритм обучается (изменяет вес/ценность причинно-следственных конструкций) и может формировать корректировки к предлагаемому перечню рекомендаций (идеальной модели), а также сигнализировать об отклонении от идеальной модели, включая наличие недостающих или избыточных запросов. Предварительно добытые данные и данные из хранилища (соответствующие субсущности в карточках внешних информационных систем) используются для расчета рисков работ из запрашиваемого перечня, а также для оценки их влияния на перевозимый объем пассажиропотока и общей оценки технического состояния.

На четвертом этапе все добытые данные совместно с другими данными, полученными в результате работы сервиса *OLAP*-кубы, расставляются на витринах для дальнейшего использования или корректировки пользователями, а также агрегации в виде совмещенной структуры и перечня работ посредством сервиса обогащения данных. На основе данных, размещенных пользователями в соответствующих сущностях/субсущностях (карточки бюджета и ресурсов) внешних информационных систем сервисами внутреннего обрабатывающего контура выполняется расчет обеспеченности (формирование ресурсных матриц по видам технических воздействий и структурных подразделений, а также сводных календарных планов, агрегированных из соответствующих планов подразделений) и расчет стоимости запрашиваемого перечня работ (сводные сметы, технико-коммерческие предложения, финансовые модели по видам работ различной агрегации в зависимости от способа группировки).

В свою очередь пользователями внешней информационной системы в соответствующих создаваемых сущностях/субсущностях на этапе пять фиксируется

использование имеющихся (резервирование доступных) или заказ недостающих ресурсов в обеспечение совмещенной структуры и перечня работ. Пользователи внешних информационных систем также формируют финансово-экономические показатели (бюджеты, нормативы и т. п.) соответствующих технических воздействий по видам работ и объектов учета (элемент, подсистема, эскалатор, станция, дистанция, хозяйство) в соответствующих сущностях/субсущностях.

Во внутренней информационной системе на последующих, с шестого по восьмой, этапах обеспеченная и расцененная совмещенная структура, и перечень работ перемещаются в сервис специалиста по работе с данными, где трансформируются в консолидированную отчетность (сводки) для последующего отображения потребителю через средства поддержки принятия решений и подготовки различного вида отчетности и дашбордов, визуализирующих процессы внутри хозяйства, посредствам интегрированных *BI* инструментов через пользовательский интерфейс, реализованный в клиентской части сервиса "Личный кабинет". Таким образом, потребитель с соответствующими правами доступа или иное лицо, ответственное за внесение изменений в сервисе "Личный кабинет", имеет возможность при соответствующем наборе входных запросов на техническое воздействие, визуализированных в виде консолидированной и *BI* отчетности, утвердить только определенную его часть, оставив другую часть неутвержденной, при этом при последующей подготовке рекомендации на восстановление данный факт будет учтен и доведен до исполнителя, который формирует запрос. Таким образом, в динамическом режиме в зависимости от ранее установленной поведенческой истории потребителя и исполнителей, отвечающих за формирование потребностей, будет меняться модель технического обслуживания и ремонта, тем самым для части элементов может быть принято решение перейти на модель по состоянию или иные.

Пользователям, назначенным в субсущностях во внешних информационных системах на те или иные виды технического воздействия, на девятом этапе, поступает на исполнение утвержденный перечень работ с плановыми значениями. Динамика исполнения и результаты работ (фактические значения) фиксируются пользователями в соответствующих сущностях/субсущностях — карточках (электронных документах). Зафиксированные данные во внешней информационной системе обновляют данные в субсущностях/сущностях обеспечивающих служб и структурных подразделений инициирующих запрос работ. Также по результатам этого этапа в «идеальную» модель (этап 0) могут быть внесены изменения посредством обновления.

Функционирование предложенной архитектуры представлено на рис. 4 и 5 и описано в табл. 2.

На рис. 5 обозначено: ***FE*** — *front-end* — клиентская часть; ***BE*** — *back-end* — серверная часть; ***DB*** — *Database* — работа с базами данных. **ОПР** — обеспеченный перечень работ; **УПР** — утвержденный перечень работ; **ЗПР** — запрашиваемый перечень работ; **РПР** — рекомендованный перечень работ. **Сервис *Data Access Management*** — сервис управления доступом и балансировки данных: **УД** — программно-аппаратный комплекс предназначенный для организации эффективной защиты периметра сети (к примеру ПАК «Рубикон»); *NLB* (*Network Load Balancing L4*) — сервис распределение приходящих запросов на несколько физических или виртуальных узлов серверов; *ALB* (*Application Load Balancing L7*) — сервис для распределения нагрузки по сервисам и приложениям; *API getway* — шлюз между пользователями и сервисами через *API*; *AUT* — модуль аутентификации пользователя. **Сервис *Data Pipeline*** — конвейер подготовки данных в составе: *ELT* — сервис работы с данными; *DL* (*Data Lake*) — хранилище сырых данных; *ETL* — сервис работы с данными; *DWH* (*Data Warehouse*) – хранилище подготовленных данных; *OLAP*-кубы — многомерный массив данных; *DM* (*Data Mart*) — множество тематических баз данных. **Сервис *Master Data Management*** —сервис управления мастер-данными и нормативно-справочной информацией в составе: КР — сервис контроля работ (управление качеством выполненных работ через балльную оценку); ЭЗ — сервис эталонных значений (управление мастер-данными); ДД — сервис дополнительных данных (управление нормативно-справочной информацией). **Сервис *Data mining*** — сервис интеллектуального анализа данных в составе: АП — сервис анализа подготовленной параметрической информации; ОР — сервис расчета остаточного ресурса; ПР — сервис нейросетевого прогнозирования; Рек — сервис подготовки рекомендаций; Рис — сервис управления рисками; ОЦ — сервис подготовки оценок. **Сервис *Data enrichment*** — сервис насыщения данных новой информацией для повышения их ценности для анализа в составе: СПР — сервис совмещения перечня работ; РОР — сервис расчета обеспеченности ресурсами; РС — сервис расчета стоимости. **Сервис *Data Steward*** — сервис управлении данными для принятия управленческих решений в составе: КО — сервис подготовки консолидированной отчетности; *DSS/EIS/BI* — сервис поддержки принятия решения и *BI* инструментов; АС ЛК — автоматизированная система "Личный кабинет" (серверный компонент); ЛК — сервис "Личный кабинет" (клиентский компонент — *UX/UI* дизайн).

Таким образом, в результате совместного использования описанных в табл. 2, микросервисов формируется экосистема, способная к развитию и адаптации к различным изменениям как ИТ-ландшафта, так и бизнес-требований.

DOI: https://doi.org/10.17816/2074-0530-109890

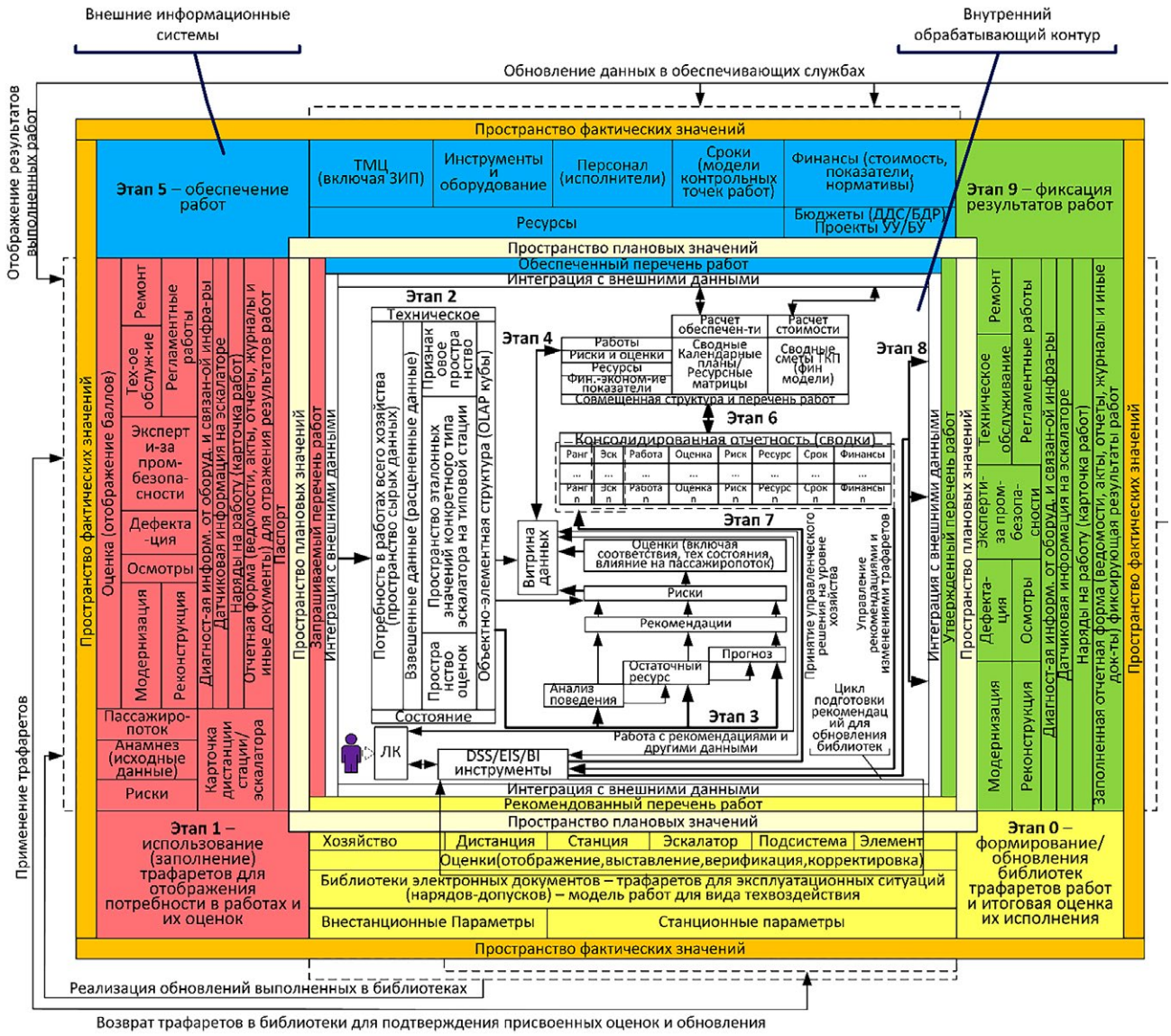

Рис. 4. Схема ИТ-ландшафта информационной системы.
Fig. 4. The scheme of the IT landscape of the information system.

ЗАКЛЮЧЕНИЕ

В качестве основного результата работы следует отметить, что из многообразия существующих сегодня паттернов разработки программного обеспечения для планирования и контроля проведения технического обслуживания и ремонта эскалаторного хозяйства метрополитена выбрана микросервисная архитектура, обеспечивающая доставку, обогащение, агрегацию и представление данных, позволяющая динамично уточнять задаваемую на начальном этапе «идеальную» модель технического состояния эскалаторного хозяйства посредством использования зафиксированного накопленного эксплуатационного опыта и изменения основных показателей эффективности, формируя возможность подстройки системы технического обслуживания и ремонта под изменяющиеся внутренние и внешние условия.

В работе также установлено, что базовым параметром, с которым работает предлагаемое программное обеспечение, является использование собственных производственных ресурсов (их планируемые, зарезервированные и нормативные значения), задействованных в операционной и проектной деятельности.

Обобщая изложенный материал, отметим, что последовательность шагов алгоритма работы предлагаемого программного обеспечения включает: определение объектов управления (к примеру, экспертиза, дефектация, обслуживания, ремонт и прочее) для построения реестров (перечней) на основе данных об использовании собственных производственных ресурсов за отчетный период (иными словами фактическая трудоемкость и работы, на которые она списана (реестр депонирования) коррелируется с планируемыми, зарезервированными

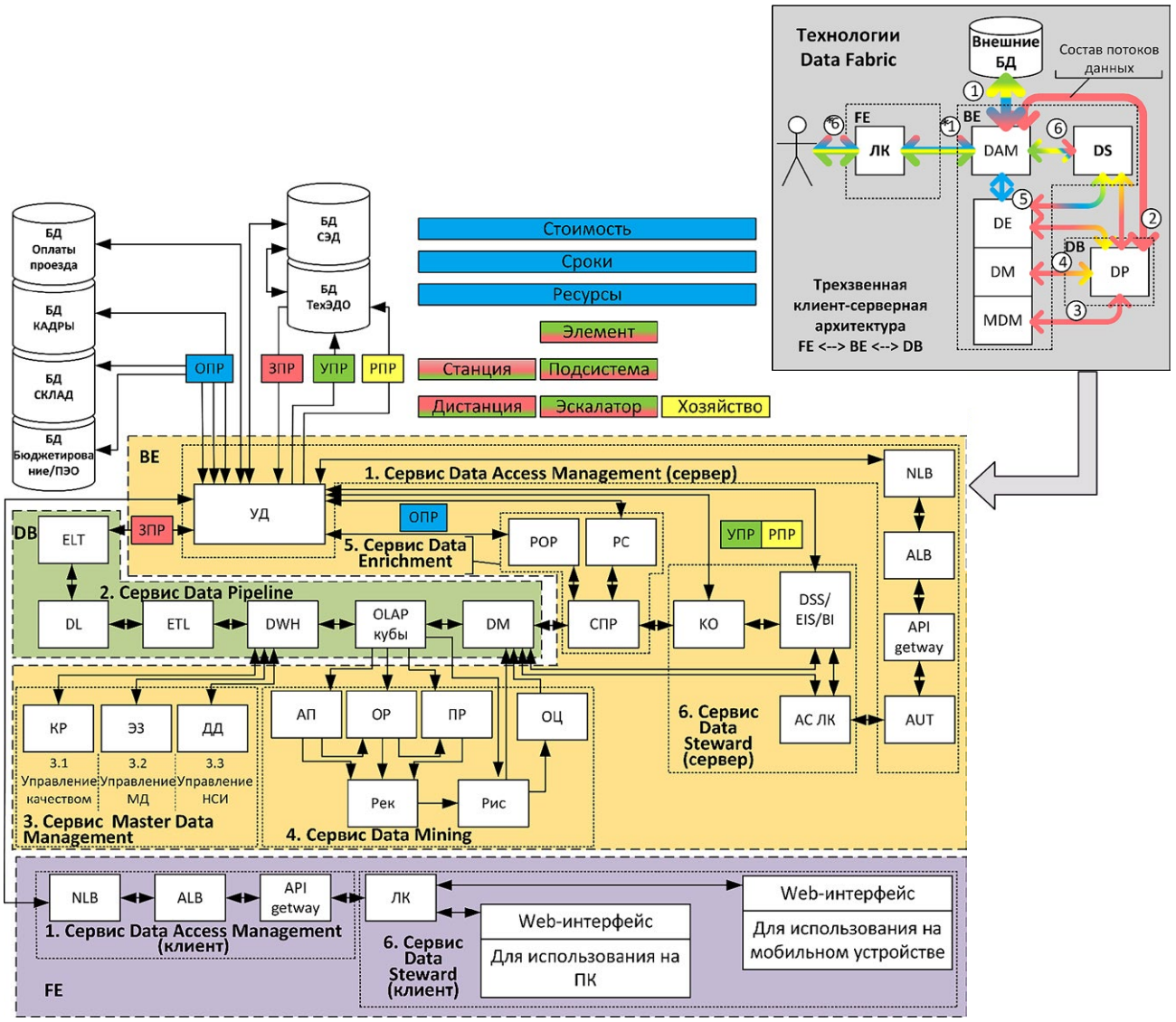

Рис. 5. Структурная схема программного обеспечения.
Fig. 5. The block diagram of the software.

БД СЭД (Архив) содержит данные о: регистрации, согласовании, распределении и хранении организационно-распорядительной, технической, производственной, эксплуатационной документации; поиске документов по атрибутам; создании документов по шаблонам; БД оплаты проезда содержит данные об объемах перевозимого пассажиропотока эскалаторного хозяйства, (в т. ч. по каждому станционному выходу);

БД КАДРЫ (Персонал) содержит данные о: списочном составе (ФИО и т.п.); квалификации (грейды); месте приписки (подразделение); должностных обязанностях; графике недоступности персонала (отпуск/болезнь/учеба);

БД СНАБЖЕНИЕ/СКЛАД (ТМЦ/инструмент/оборудование) содержит данные о: оформлении и статусе заявок на приобретение; резервирование/отпуске; учете движения на складах и в подотчете (в т.ч. остатки);

БД БЮДЖЕТИРОВАНИЕ/ПЭО (Фин/эконом характеристики) содержит данные о: проектах управленческого учета/бухгалтерского учета по заведованию; эконом. нормативах (стоимость н/ч, % накладных расходов и т.п.) по видам работ; бюджетах по подразделениям/видам деятельности (в т.ч. структура затрат/поступления (выручка)/графики их реализаций); экономическим характеристикам (внутренняя доходность, рентабельность и т.п.); характеристикам инвестиционной деятельности (потребность в инвестировании, длительность инвестирования, срок возврата инвестиций, статьи затрат по инвестициям, срок окупаемости (рентабельности));

БД ТехЭДО (Техническое состояние) содержит данные об: объектах эскалаторного хозяйства и связанной с ними инфраре (тотс/раб, отказ/поломка, список ЗПЧ и т.п.); данные о проведенных тех. воздействиях (журналы, ведомости и иное); данные о тех состоянии (журналы, отчеты и иное); паспорта на каждый элемент/функц. подсистемы и эскалатор целом; план-график работ; и др. техническая информация.

Таблица 2. Описание последовательности этапов работы программного обеспечения
Table 2. Description of a sequence of stages of the software operation

Процесс	Этап	Содержание	Данные	Модули	Сервисы
Внешний	Этап 0	Формирование/обновление библиотек трафаретов и итоговая оценка исполнения работ	Рекомендованный перечень работ	Внешние информационные системы	Сервисы внешних информационных систем
	Этап 1	Использование (заполнение) трафаретов для отображения потребности в работах и их оценок	Запрашиваемый перечень работ		
Внутренний	Этап 2	Конвейер подготовки данных из запрашиваемой потребности в работах	Запрашиваемый перечень работ	*ELT, DL, ETL, DWH, OLAP*-кубы, *DM*	Сервис *Data Pipeline*
	Этап 3	Аналитическая обработка и оценка подготовленных данных	Запрашиваемый перечень работ	КР, ЗЗ, ДД, АП, ОР, ПР, Рек, Рис, ОЦ	Сервис *Master Data Management*, Сервис *Data Mining*
	Этап 4	Расстановка данных в витрине для совмещения потребности в работах с результатами аналитической обработки и предоставления пользователям	Запрашиваемый перечень работ	СПР, РОР, РС	Сервис *Data Enrichment*
Внешний	Этап 5	Обеспечение работ	Обеспеченный перечень работ	Внешние информационные системы	Сервисы внешних информационных систем
Внутренний	Этап 6	Формирование консолидированной отчетности (сводки) по хозяйству	Обеспеченный перечень работ	КО	Сервис *Data Steward*
	Этап 7	Принятие управленческого решения по составу работ и подготовка рекомендаций по обновлению библиотек	Утвержденный перечень работ	*DSS, EIS, BI, AC* ЛК	
	Этап 8	Выдача на исполнение утвержденного перечня работ	Утвержденный перечень работ		
Внешний	Этап 9	Фиксация результатов работ	Утвержденный перечень работ	Внешние информационные системы	Сервисы внешних информационных систем

и нормативными значениями); дополнение сведений о фактической трудоемкости оперативными данными из различных источников об объектах управления размещенных в реестре, их предварительная обработка, в результате которой фиксируется текущее состояние; определение отклонения текущего состояния объектов управления от целевого, подсвечивание проблем и «узких» мест; представление сводных данных потребителям для дальнейшего использования.

Таким образом, предлагаемое программное обеспечение предназначено для регулярного использования в качестве инструмента мониторинга и контроля реестра объектов управления, создающих и не создающих ценность для компании (обеспечивающих/не обеспечивающих транспортировку пассажиропотоков).

Целью предлагаемого программного обеспечения является оценка степени управляемости объектов управления из реестра и отображение эффективности использования собственных ресурсов на их реализацию.

Основными задачами, решаемыми предлагаемым инструментом, являются наглядное отображение текущего состояния (при необходимости накопительные итоги) объектов управления, попавших в реестр за отчетный период, обеспечивающее общее представление

о их статусе управления и идентификации проблемных участков; фокусировка внимания лица, принимающего решение (ЛПР), или иного потребителя на выделенных текущих значениях ключевых параметров объекта управления и метриках их измерения, т. к. текущее состояние объекта управления влияет на величину *KPI* ЛПР; отслеживание влияния задаваемых плановых показателей верхнего уровня и привил меппинга данных между используемыми информационными системами — источниками на другие показатели; использование источников первичных данных, фиксирующих ключевые параметры и их метрики; детализация информации (при необходимости до исполнителя).

Реализация предлагаемого программного обеспечения возможна с использованием стандартных продуктов, широко представленных на рынке, что сокращает время на его разработку и внедрение.

Полученные результаты могут являться одним из вариантов реализации концепции цифровой трансформации системы технического обслуживания и ремонта эскалаторного хозяйства метрополитена.

ДОПОЛНИТЕЛЬНО

Вклад авторов. В.А. Попов — поиск публикаций по теме статьи, редактирование текста и изображение рукописи, экспертная оценка, утверждение финальной версии; В.В. Еланцев — написание текста рукописи и создание изображений. Авторы подтверждают соответствие своего авторства международным критериям *ICMJE* (все авторы внесли существенный вклад в разработку концепции, проведение исследования и подготовку статьи, прочли и одобрили финальную версию перед публикацией).

Конфликт интересов. Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.

Источник финансирования. Авторы заявляют об отсутствии внешнего финансирования при проведении исследования.

ADDITIONAL INFORMATION

Authors' contribution. V.A. Popov — search for publications, editing the text and images of the manuscript, expert opinion, approval of the final version; V.V. Elantsev — writing the text and creating images of the manuscript. All authors made a substantial contribution to the conception of the work, acquisition, analysis, interpretation of data for the work, drafting and revising the work, final approval of the version to be published and agree to be accountable for all aspects of the work.

Competing interests. The authors declare that they have no competing interests.

Funding source. This study was not supported by any external sources of funding.

СПИСОК ЛИТЕРАТУРЫ

1. Итоги социально-экономического развития Санкт-Петербурга за январь-май 2022. Выпуск 5. [интернет]. Санкт-Петербург: Комитет по экономической политике и стратегическому планированию Санкт-Петербурга, 2022. Дата обращения: 14.08.2022. Доступ по ссылке: https://cedipt.gov.spb.ru/media/uploads/userfiles/2022/07/04/СПРАВКА_ЧП_янв-май_2022_электронный_вариант.pdf

2. Бухгалтерская (финансовая) отчетность метрополитена за первый квартал 2022 года: официальный сайт [интернет]. Санкт-Петербург: ГУП «Петербургский метрополитен», 2022. Дата обращения: 14.08.2022. Доступ по ссылке: http://www.metro.spb.ru/uploads/document/fin_otch_1kv2022.pdf

3. Корнеева А.В., Корнеев Г.У. Бизнес-процессы: от ценности к прибыли // Вестник Алтайской академии экономики и права. 2021, № 7(2). С. 168–175. doi: 10.17513/vaael.1795

4. Варзунов А.В., Торосян Е.К., Сажнева Л.П. Анализ и управление бизнес-процессами. Учебное пособие. Санкт-Петербург: Университет ИТМО, 2016. Дата обращения: 13.08.2022. Доступ по ссылке: https://books.ifmo.ru/file/pdf/2017.pdf

5. Павлов А.В. Архитектура вычислительных систем. Санкт-Петербург: Университет ИТМО, 2016. Дата обращения: 13.08.2022. Доступ по ссылке: https://books.ifmo.ru/file/pdf/2074.pdf

6. Хорошевский В.Г. Архитектура вычислительных систем. Учебное пособие. М.: Изд-во МГТУ им. Н.Э. Баумана, 2008. Дата обращения: 13.08.2022. Доступ по ссылке: https://nsu.ru/xmlui/bitstream/handle/nsu/935/khor32.pdf

7. Дандан Р. Методические материалы по формированию карты контекстов // Международный журнал гуманитарных и естественных наук. 2022. Т. 5–2(68). С. 21–25. doi: 10.24412/2500-1000-2022-5-2-21-25

8. Сиразетдинов Р.Р., Белоус Д.В. Архитектура информационных систем // Технические средства связи. 2020. № 3. С. 65–68.

9. Воробьев А.В., Воробьева Г.Р. Модель информационного взаимодействия элементов многоуровневой системы цифровых двойников // Информатика и автоматизация. 2021. Т. 20, № 3. С. 530–561. doi: 10.15622/ia.2021.3.2

10. Плетнев А.В. Выбор технологического стека для IT-проекта // Интернаука: электрон. научн. журнал. 2021. № 36(212). doi: 10.32743/26870142.2021.36.212.303217

11. Городничев М.Г., Полонский Р.В. Оценка возможности использования микросервисной архитектуры при разработке пользовательских интерфейсов клиент-серверного программного обеспечения // Экономика и качество систем связи. 2020. № 3(17). С. 33–43. Режим доступа: http://nirit.org/wp-content/uploads/2020/09/33-43.pdf Дата обращения: 13.08.2022.

12. Ричардсон К. Микросервисы. Паттерны разработки и рефакторинга. Санкт-Петербург: Питер, 2019.
13. Попов В.А., Еланцев В.В. К вопросу повышения эффективности и безопасности эксплуатации тоннельных эскалаторов метрополитена. Концепция цифровой трансформации системы ТОиР // Системы автоматизированного проектирования на транспорте: материалы IX Международной науч.-практ. конф. студентов, аспирантов и молодых ученых (Санкт-Петербург, 27–28 апреля 2021 г.). Санкт-Петербург: Изд-во ФГБОУ ВО ПГУПС, 2021. С. 91–96.
14. Шинкарев А.А. Ретроспектива развития веб-технологий в создании корпоративных информационных систем // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. 2020. Т. 20, № 4. С. 14–21. doi: 10.14529/ctcr200402
15. Симанков В.С., Шарай В.А. Программное и аппаратное обеспечения подсистем интеллектуального ситуационного центра // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. 2021. № 3(286). С. 63–72. doi: 10.53598/2410-3225-2021-3-286-42-54
16. Распоряжение Правительства Российской Федерации № 80-р от 25 января 2018 г. «О развитии проектной деятельности в Правительстве России» (с изменениями и дополнениями). Режим доступа: http://government.ru/docs/31211/ Дата обращения: 13.08.2022.
17. Методические рекомендации Правительства Российской Федерации «Методические рекомендации по организации проектной деятельности в федеральных органах исполнительной власти» № 1937п-П6 от 12 марта 2018 г. (с изменениями и дополнениями) Режим доступа: http://government.ru/info/31672/ Дата обращения: 13.08.2022.
18. Методическое пособие «Функционирование проектных офисов» (с изменениями и дополнениями) [интернет] М.: Центр проектного менеджмента, 2022. Дата обращения: 13.08.2022. Доступ по ссылке: https://pm.center/bazaznaniy/document/metodicheskoe-posobie-funktsionirovanie-proektnykh-ofisov/

REFERENCES

1. Results of the socio-economic development of St. Petersburg for January-May 2022. Iss. 5. [internet]. St. Petersburg: Committee for Economic Policy and Strategic Planning of St. Petersburg; 2022. (in Russ). [cited 2022 Aug 14] Available from: https://cedipt.gov.spb.ru/media/uploads/userfiles/2022/07/04/СПРАВКА_ЧП_янв-май_2022_электронный_вариант.pdf
2. Accounting (financial) statements of the metro for the first quarter of 2022: official website [internet]. St. Petersburg: State Unitary Enterprise "Petersburg Metro"; 2022. (in Russ). [cited 2022 Aug 14] Available from: http://www.metro.spb.ru/uploads/document/fin_otch_1kv2022.pdf
3. Korneeva AV, Korneev GU. Business processes: from value to profit. *Vestnik Altayskoy akademii ekonomiki i prava.* 2021;7(2):168–175. (in Russ). doi: 10.17513/vaael.1795
4. Varzunov AV, Torosyan EK, Sazhneva LP. *Analysis and management of business processes.* St. Petersburg: ITMO; 2016. (in Russ). [cited 2022 Aug 14] Available from: https://books.ifmo.ru/file/pdf/2017.pdf
5. Pavlov AV. *Architecture of computing systems.* St. Petersburg: ITMO; 2016. (in Russ). [cited 2022 Aug 14] Available from: https://books.ifmo.ru/file/pdf/2074.pdf
6. Khoroshevsky VG. *Computing architecture. Tutorial.* Moscow: Izd-vo MGTU im. H.E. Baumana; 2008. (in Russ). [cited 2022 Aug 14] Available from: https://nsu.ru/xmlui/bitstream/handle/nsu/935/khor32.pdf
7. Dandan R. Methodological materials for the formation of a map of contexts. *International Journal of the Humanities and Natural Sciences.* 2022;5–2(68):21–25. (in Russ). doi: 10.24412/2500-1000-2022-5-2-21-25
8. Sirazetdinov RR, Belous DV. Information systems architecture. *Tekhnicheskie sredstva svyazi.* 2020;3:65–68. (in Russ).
9. Vorobyov AV, Vorobieva GR. Model of information interaction of elements of a multilevel system of digital twins. *Informatika i avtomatizatsiya.* 2021;20(3):530–561. (in Russ). doi: 10.15622/ia.2021.3.2
10. Pletnev AV. Choosing a technology stack for an IT project. *Internauka: elektron. nauchn. zhurnal.* 2021;36(212). (in Russ). doi: 10.32743/26870142.2021.36.212.303217
11. Gorodnichev MG, Polonsky RV. Evaluation of the possibility of using microservice architecture in the development of user interfaces for client-server software. *Ekonomika i kachestvo sistem svyazi.* 2020;3(17):33–43. (in Russ). [cited 2022 Aug 14] Available from: http://nirit.org/wp-content/uploads/2020/09/33-43.pdf
12. Richardson K. *Microservices. Patterns of development and refactoring.* St. Petersburg: Piter; 2019. (in Russ).
13. Popov VA, Elantsev VV. On the issue of improving the efficiency and safety of operation of underground tunnel escalators. The concept of digital transformation of the maintenance and repair system. In: *Sistemy avtomatizirovannogo proektirovaniya na transporte: materialy IX Mezhdunarodnoy nauch.-prakt. konf. studentov, aspirantov i molodykh uchenykh* (St. Petersburg, April 27–28, 2021). St. Petersburg: Izd-vo FGBOU VO PGUPS; 2021:91–96. (in Russ).
14. Shinkarev AA. Retrospective of the development of web technologies in the creation of corporate information systems. *Vestnik Yuzhno-Uralskogo gosudarstvennogo universiteta. Seriya: Kompyuternye tekhnologii, upravlenie, radioelektronika.* 2020;20(4):14–21. (in Russ). doi: 10.14529/ctcr200402
15. Simankov VS, Sharay VA. Software and hardware of the subsystems of the intelligent situational center. Vestnik Adygeyskogo gosudarstvennogo universiteta. Seriya 4: Estestvenno-matematicheskie i tekhnicheskie nauki. 2021;3(286):63–72. (in Russ). doi: 10.53598/2410-3225-2021-3-286-42-54
16. Decree of the Government of the Russian Federation No. 80-r dated January 25, 2018 "On the development of project activities in the Government of Russia" (with amendments and additions). (in Russ). [cited 2022 Aug 14] Available from: http://government.ru/docs/31211/
17. Methodological recommendations of the Government of the Russian Federation "Guidelines for the organization of project

activities in federal executive bodies" No. 1937p-P6 dated March 12, 2018 (as amended). (in Russ). [cited 2022 Aug 14] Available from: http://government.ru/info/31672/ Дата обращения: 13.08.2022.
18. Methodological guide "Functioning of project offices" (with changes and additions) [internet] Moscow: Tsentr proektnogo menedzhmenta; 2022. (in Russ). [cited 2022 Aug 14] Available from: https://pm.center/bazaznaniy/document/metodicheskoe-posobie-funktsionirovanie-proektnykh-ofisov/

ОБ АВТОРАХ

*** Еланцев Валентин Валентинович,**
аспирант кафедры «Наземные транспортно-технологические комплексы»;
адрес: Российская Федерация, 190031, Санкт-Петербург, Московский проспект, д. 9;
ORCID: https://orcid.org/0000-0003-1731-5626;
eLibrary SPIN: 9667-9716;
e-mail: evv3012@gmail.com

Попов Валерий Анатольевич,
доцент, канд. техн. наук,
доцент кафедры «Наземные транспортно-технологические комплексы»;
ORCID: https://orcid.org/0000-0003-2635-5427;
eLibrary SPIN: 2418-7152;
e-mail: a vpopov_58@mail.ru

*** Автор, ответственный за переписку**

AUTHORS' INFO

*** Valentin V. Elantsev,**
postgraduate of the Mechanical Handling and Road Building Machines Department;
address: 9 Moskovsky ave., 190031 Saint Petersburg, Russian Federation;
ORCID: https://orcid.org/0000-0003-1731-5626;
eLibrary SPIN: 9667-9716;
e-mail: evv3012@gmail.com

Valery A. Popov,
Associate Professor, Dr. Sci. (Tech.),
Associate Professor of the Mechanical Handling and Road Building Machines Department;
ORCID: https://orcid.org/0000-0003-2635-5427;
eLibrary SPIN: 2418-7152;
e-mail: a vpopov_58@mail.ru

*** Corresponding author**