ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 13**, № 4(55), с. 93–109

научная статья ИНФОРМАЦИОННЫЕ СИСТЕМЫ В МЕДИЦИНЕ

УДК 61:004.652

10.25209/2079-3316-2022-13-4-93-109

Опыт импортозамещения в медицинской информационной системе «Интерин PROMIS Alpha»

Дмитрий Владимирович **Белышев**✉

Институт программных систем им. А. К. Айламазяна РАН, Веськово, Россия

✉*belyshev@interin.ru* *(подробнее об авторе на с. 107)*

Аннотация. Импортозамещение в части систем управления базами данных является насущной задачей, стоящей перед многими производителями информационных систем. В нашей работе мы рассматриваем опыт миграции медицинской информационной системы Интерин PROMIS из технологий Windows/Oracle в технологии на основе Linux/PostgreSQL, рекомендуемые Реестром отечественного ПО. Дается оценка двух подходов к миграции модулей информационной системы: на основе перепроектирования с использованием инструментов платформы Интерин Alpha PG и на основе автоматической конвертации собственными инструментальными средствами.

(*see abstract in English on p. 108*)

Ключевые слова и фразы: медицинская информационная система, импортозамещение, PostrgeSQL

Для цитирования: Белышев Д. В. *Опыт импортозамещения в медицинской информационной системе «Интерин PROMIS Alpha»* // Программные системы: теория и приложения. 2022. **Т. 13**. № 4(55). С. 93–109. http://psta.psiras.ru/read/psta2022_4_93-109.pdf

Введение

Задача импортозамещения в информационных технологиях является насущной и широко обсуждаемой: государство уделяет значительное внимание вопросам развития отечественной ИТ-отрасли, что проявляется в широком спектре решений: от законодательного ужесточения использования отечественного ПО до налоговых льгот, а также крупных государственных проектов и грантовой поддержки отечественных ИТ-компаний [1].

В аналитическом обзоре *TAdviser «Драйверы ИТ-импортозамещения в России»* отмечается, что вопрос импортозамещения стоял на повестке государственных компаний и предприятий ещё с конца 90-х – начала 2000-х годов, однако реальное продвижение тренд получил только в 2014 году после введения первых санкций против госсектора российской экономики. В результате 29.06.2015 был принят федеральный закон №188-ФЗ «О внесении изменений в Федеральный закон „Об информации, информационных технологиях и о защите информации“», регламентирующий создание *Реестра российского программного обеспечения* и накладывающий ограничения на госзакупки товаров и услуг иностранных государств, что дало законодательную основу для развития отечественных ИТ-технологий. Кроме того, в это же время в связи с резким ростом курса иностранной валюты переход на российское ПО стал экономически целесообразным и для коммерческих компаний. В той же публикации рассмотрены следующие направления, стимулирующие интерес к отечественному ПО:

– Какие наиболее важные события стимулировали импортозамещение ПО в 2021 году:
  - персональная ответственность за импортозамещение;
  - масштабные проекты цифровой трансформации и меры поддержки ИТ-отрасли;
  - давление регуляторов;
  - рост доверия к отечественным решениям;
  - новые партнерства в ИТ-сфере и развитие совместных решений;
  - последствия коронакризиса;
  - отраслевые особенности.
– Что простимулирует процесс импортозамещения в 2022 году:

- геополитика и санкции;
- стремление к технологическому суверенитету;
- реакция рынка и дальнейшие пути импортозамещения.

Требования законодательства и изменение рыночных условий не могли не коснуться сектора медицинских информационных систем, поскольку подавляющее большинство крупных медицинских организаций находятся в государственной собственности и автоматически подпадают под введенные ограничения. Коммерческие медицинские центры также испытывают серьезное давление, но уже в связи с возросшей стоимостью иностранных ИТ-решений из-за изменения валютного курса, а впоследствии и в связи с фактическим уходом ряда иностранных ИТ-компаний с российского рынка.

Оценка рисков и меры по их купированию были заблаговременно предприняты группой компаний Интерин, была проведена необходимая подготовительная работа, и 01.06.2016 Медицинская информационная система Интерин PROMIS *включена в реестр отечественного ПО*. Вместе с тем, хотя программное обеспечение и является собственной разработкой, но среда его функционирования на тот момент включала иностранное системное ПО (в первую очередь это касалось операционной системы и СУБД), что потребовало дальнейшей активной работы.

Если рассмотреть развитие МИС «Интерин PROMIS» в исторической ретроспективе, то в 1990-х годах, когда начиналось формирование рынка МИС, отечественных или даже приемлемых опенсорсных решений в части СУБД для построения крупных корпоративных информационных систем фактически не было (тот же PostgreSQL появился только 1996 году). На начало 2000 годов по представленным *Алексеем Резниченко в ITWeek данным компаний IDC и Dataquest* СУБД Oracle занимала 42% мирового рынка, вслед за ней шли IBM (20%) и Microsoft (8%). В то же время по данным *обзора Tadviser* в России на 2011 год Oracle занимала все 70% рынка.

Решающим фактором выбора платформы для Интерин PROMIS стал якорный заказчик – Медицинский центр Банка России, для которого впервые в 1996 году была создана МИС семейства *Интерин*. Поскольку для банковской сферы использование Oracle как ранее, так и до сих пор является наиболее распространенным решением, МИС также использовала эту СУБД.

Следует признать, что за почти 30-летний опыт эксплуатации связки СУБД OracleDB и средств построения пользовательского

интерфейса Oracle Developer доказали высокую надежность, гибкость и эффективность в решении задач построения информационных систем крупных клиник, которые эксплуатируются с момента запуска и по сей день.

Вместе с тем, глубокая интеграция средств разработки с СУБД, которая существенно упрощает разработку и делает более эффективной обработку данных, в то же время оказалась и существенным препятствием в изменении архитектуры, поскольку двухзвенная технология «родного толстого клиента», предоставленного Oracle Developer, не предусматривает никаких альтернативных решений, а бизнес-логика приложений оказалась распределенной между клиентской и серверной частями системы. Причем, особенностью СУБД Oracle является наличие развитых средств разработки бизнес-логики непосредственно на стороне СУБД. Это дает большие преимущества за счет более тесной связи данных и средств их обработки, но обратной стороной является полная зависимость от среды исполнения. В результате преимущества работы в интегрированной среде обернулись серьезными препятствиями при попытке перехода на другие технологии, поскольку все процессы оказались плотно завязаны друг на друга: клиент мог работать только с имеющимся сервером, поэтому безальтернативным вариантом первого шага трансформации была замена клиентского ПО.

1. Первый этап: миграция на «тонкого клиента»

Выше был рассмотрен ряд обстоятельств, подталкивающих МИС к смене технологической базы, однако, не только требования регуляторов и стоимость импортных решений определяли необходимость перехода. В значительной степени, развитие рынка, рост бизнес-требований заказчиков, давление со стороны конкурентов заставляло искать новые, прежде всего клиентские интерфейсные решения. Так, если в 2000-х годах оконные интерфейсы Oracle Forms со статичными формами ввода и списками ещё могли считаться более-менее приемлемыми, то постепенно с развитием веб-технологий, предоставляющих гибкие адаптивные интерфейсы, существенно изменились ожидания пользователей. Также нужно упомянуть привязку Oracle Forms 6i к ОС Windows, что, в свою очередь, вводило ограничения на импортозамещение клиентской ОС, а это оказалось чувствительно как экономически, так и в части требований регуляторов по использованию иностранного ПО в госучреждениях.

В результате всех обстоятельств единственным выходом являлась замена клиентского интерфейса, причем направления изменений были далеко не однозначными, так как на момент начала работ типовая МИС Интерин насчитывала более 2 000 клиентских модулей, более того, на всех проектах дополнительно присутствовали свои модификации форм и отчетов. Требовалось выполнить миграцию за максимально короткое время с минимальными затратами и без потери функциональных возможностей. Рассматривался спектр решений:

– Остаться в инфраструктуре Oracle, что в свою очередь можно было сделать несколькими путями:
  • перенести имеющиеся модули средствами конвертации в Oracle Fusion Middleware, функционирующую в Oracle WebLogic Server, и продолжить разработку в более-менее близкой технологии;
  • выполнить перенос в Oracle Application Development Framework (ADF) с дальнейшей разработкой в J2EE;
  • переписать формы в Oracle Apex, оставшись с PLSQL в качестве основного языка бизнес-логики.
– Уйти от использования инфраструктуры Oracle, что означало бы полное перепроектирование и разработку «с нуля» всех клиентских модулей.

Принятие того или иного решения было очень непростой задачей, поскольку от него зависело настоящее и будущее как программного продукта, так и собственно компании. Еще в 2011 году, задолго до начала активных действий по импортозамещению, были выполнены первые шаги по смене клиентского софта применением типовых решений, которые предлагал Oracle (в то время ни о каких санкциях речи не шло), и была выполнена конвертация значительной части клиентских форм в Oracle Fusion Middleware, что позволило запускать интерфейсы «как есть» в апплетах браузера. Технически перенос показал работоспособность, но качество решений лишь уменьшилось, поскольку все имеющиеся ограничения, которые были у старых форм, были просто перенесены в новую среду. Тестирование Oracle ADF показало существенный рост требований к коллективу разработки, поскольку требовалась миграция в Java и перенос в нее унаследованного программного кода клиентских форм. Исследование Oracle Apex показало, что это достаточно ограниченный инструмент, позволяющий довольно быстро реализовывать небольшие и сравнительно мало

насыщенные интерфейсы, тогда как рабочие места в медицинской системе требуют крайне разнообразного функционала. Вместе с тем, достаточная простота и возможность остаться в привычных технологиях позволили реализовать несколько специализированных проектов на Oracle Apex, что дало определенный опыт в переносе в веб имеющихся решений и позволило его использовать в последующей работе.

Эксперименты и отдельные разработки, связанные с поиском новых интерфейсных решений в среде Oracle, шли до 2014 года, но уже в 2015 году была начата работа по созданию собственной программной платформы, которая в итоге получила название *«Платформа Интерин IPS»*. Ядром Платформы стали технология конструктивного синтеза пользовательских интерфейсов Web-приложений [2] и JSON-хранилища данных [3, 4]. Помимо технологических аспектов, значительное место в проектировании и реализации платформы занимали организационные и методические вопросы: как обеспечить наиболее плавную бесшовную миграцию информационной системы, коллектива разработчиков и функционирующих проектов в новую систему, не растеряв ни функциональных возможностей системы, ни компетенций разработчиков, ни заказчиков. Еще одной особенностью данного процесса являлось то, что создание платформы пришлось на период экономического кризиса 2014–2015 годов и послекризисный период, что делало все принимаемые решения ещё более чувствительными.

Уже через год после начала разработки Платформы IPS была начата миграция клиентского прикладного программного обеспечения с «толстого» клиента, и в 2017 году была создана первая версия МИС *Интерин PROMIS Alpha* – результат переноса клиентского ПО из Oracle Developer в собственную вебплатформу. В том же году было начато первое внедрение обновленной МИС. На этом процесс миграции не остановился, МИС Интерин PROMIS Alpha продолжала расширять состав своих функций, постепенно превзойдя функциональные возможности МИС предыдущих версий и достигнув количества клиентских модулей более 5000.

Оценивая пройденный путь по модернизации технологий разработки и фактического переписывания клиентской части МИС, можно отметить, что во многом реализованная схема была вынужденной, а принимаемые решения высокорискованными. Мы пошли на максимальный риск, связанный с созданием собственной платформы и перепроектированием всей системы под нее, но и потенциальный

эффект ожидался максимальным. Результаты внедрений новой системы имеют хорошие показатели: как интерфейсные, так и функциональные возможности существенно превзошли ранее достигнутые параметры, существенно выросла эффективность разработки и снизилась стоимость сопровождения за счет интеграции непосредственно в веб-интерфейс пользователя средств конструирования всех компоненты системы, появились удобные средства контроля и отладки приложений, управления обновлениями и версионностью софта – это и многое другое является следствием использования собственных инструментальных средств и среды исполнения, реализованных под собственные нужды.

2. Второй этап: миграция на СУБД PostgreSQL

2.1. Трудности перевода

Очевидно, что группа компаний Интерин не рассматривает импортозамещение инфраструктурного ПО как соответствие модному тренду, а принимает его как объективную рыночную необходимость, и переход в веб, таким образом, не являлся конечной целью решаемой задачи. Новые интерфейсные возможности и способы организации бизнес-логики и обработки данных дали существенный импульс развитию МИС и позволили предложить клиентам новые функциональные возможности, отвечающие актуальным потребностям рынка. Вместе с тем, задача перехода на отечественную СУБД стала ещё более актуальной, поскольку ни стоимость владения, ни новые геополитические вызовы не способствуют тому, чтобы можно было рассматривать Oracle как надежного партнера для построения информационных систем в России. Переход к трехзвенной архитектуре в значительной степени отвязал клиентские модули от СУБД, что открыло возможности для её замещения.

В отличии от выбора клиентской технологии, выбор альтернативы СУБД Oracle более очевиден, так как наиболее близким по возможностям является PostgreSQL, имеющая зарегистрированную в Реестре отечественного ПО версию PostrgresPro. Наиболее существенной в нашем случае особенностью PostgreSQL является наличие полноценных средств написания серверных процедур на языке PL/pgSQL, близком по синтаксису к Oracle PL/SQL. Это очень важный фактор, поскольку в нашем случае только количество серверного программного кода в хранимых процедурах СУБД составляет порядка 1.5 млн.

строк, в сумме же с программным кодом, исполняемым из клиентских модулей, объем оценивается в 2 млн. строк кода.

Как и в случае с клиентом приходится рассматривать разные варианты решения задачи миграции:

1. автоматическая миграция при помощи универсальных инструментов конвертации базы данных с последующей правкой получившегося кода;
2. выборочная миграция базы данных специализированными инструментами конвертации процедур и структур данных с последующей правкой получившегося кода;
3. полное или частичное перепроектирование базы данных с переписыванием бизнес-логики.

Учитывая, что масштаб трансформации значительный, были предприняты шаги по анализу наиболее простых и технологичных способов решения задачи. Был проведен анализ и опробованы типичные утилиты и методы конвертации СУБД [5], в 2018 году был проведен аудит системы специалистами PostgresPro для оценки стратегии миграции. Результат поиска способа механического перевода системы и мнение специалистов оказались удручающими:

> «*К сожалению вердикт неутешителен. В текущем виде мигрировать систему без изменения архитектуры будет очень сложно и рискованно. Наша рекомендация – разработка новой версии системы, ориентированной на Postgres или работу с несколькими СУБД.*»

В целом, полученные оценки оказались хоть и неприятными, но вполне ожидаемыми, так ещё весной 2016 года российский офис Oracle отправил электронные письма нескольким сотрудникам российских IT-компаний: *«Почему PostgreSQL не является аналогом СУБД Oracle»*. В сообщении указано 10 ключевых с точки зрения компании Oracle причин, которые не позволяют рассматривать PostgreSQL в качестве конкурента Oracle.

В статье «Импортозамещение: риски и иллюзии» [6], опубликованной в январе 2015 года, сразу после начала санкционного давления, приводится масса аргументов, доказывающих почему отечественные СУБД или системы с открытым кодом, не являются аналогами коммерческих систем западного производства. Вместе с тем, представители бизнеса пытаются объективно смотреть на реальное положение

дел и оценивают необходимый перенос систем как сложную, но не невозможную задачу, так в статье *«Как российский бизнес замещает СУБД Oracle и Microsoft»* «Ведомости» приводят результаты опроса руководителей крупнейших российских компаний в разделе «Трудности перевода»:

- *Теоретически PostgreSQL может решить все проблемы, но миграция может потребовать существенного пересмотра архитектуры прикладных систем, спроектированных под Oracle или Microsoft SQL Server, говорит представитель ВТБ. Поэтому банк, отмечает его представитель, использует Postgres пока только в собственной разработке нового ПО.*
- *Ренат Лашин из «Отечественного софта» говорит, что большинство корпоративных систем можно перевести на Postgres за один-два года, но в некоторых случаях систему проще разработать заново, построив её по современным принципам.*
- *Основная проблема, по словам Гольцова из «АМТ груп», заключается в том, что требуется не только перенести данные, но и переписать приложения, которые обрабатывают эти данные. Часть операций переноса данных невозможно автоматизировать, нужно перепрограммировать функции вручную, поэтому миграция может оказаться трудоемкой, говорит Гольцов.*

2.2. Перенос платформы

Невозможность автоматической конвертации базы данных потребовала более тщательного рассмотрения особенностей решаемой задачи, так в миграции серверной части системы в PostgreSQL было выделено два последовательных этапа:

(*1*) перенос Платформы Интерин IPS;

(*2*) перенос прикладных подсистем МИС.

Отметим, что для этапа переноса платформы метод миграции безальтернативно заключается в перепроектировании, поскольку системная часть должна обеспечивать максимальную эффективность работы всех инструментов, что можно сделать только учитывая особенности архитектуры СУБД. Так, ярким примером различий двух СУБД является поддержка JSON. Если в Oracle до 12 версии поддержка JSON полностью отсутствовала, и в платформе пришлось использовать собственную реализацию данного типа данных, то в PostgreSQL тип

JSONB является встроенным в СУБД, может использоваться в качестве полей таблиц, применяться в SQL-запросах, доступна индексация, имеется развитый функциональный инструментарий. В силу этих и других особенностей, в том числе с учетом уже накопленного опыта эксплуатации Платформы Интерин IPS в проектах на Oracle, в течение 2021 года было проведено перепроектирование хранилищ данных [3] и перенесен программный код платформы в PostgreSQL. Таким образом, к началу 2022 года платформа на PostgreSQL, получившая название «Интерин Alpha PG» была готова к началу прикладной разработки.

2.3. Перенос прикладных модулей МИС

Для прикладных модулей способ миграции был не такой очевидный, как для системных, поэтому было сформулировано два пути перевода, решение о применении каждого из которых соответствует сути конкретной решаемой задачи:

1. выборочная конвертация структур данных и бизнес-логики, реализованной в виде хранимых процедур Oracle (пакеты, функции, процедуры, триггеры) и программного кода, включенного в состав модулей Платформы IPS;
2. выборочное перепроектирование схемы данных и бизнес-логики с использованием технологии модульных хранилищ Платформы Интерин Alpha PG [3].

Для выполнения конвертации структур данных и процедур был разработан ряд инструментальных средств «Средства миграции O2P», позволяющих вычислять логические связи между объектами (таблицы и хранимые процедуры) в Oracle, выполнять их конвертацию в корректные структуры PostgreSQL, включающую перенос структур платформы с частичной трансляцией программного кода на PL/SQL в PL/pgSQL и JavaScript. Методика использования инструментов конвертации основывается, в первую очередь, на анализе архитектуры той или иной подсистемы. В случае, если архитектура подсистемы удовлетворяет реализации в PostgreSQL и текущим бизнес-требованиям, то как структуры данных, так и процедуры их обработки разумно конвертировать, что позволит быстро получить достаточно большой объем кода в PostgreSQL и в дальнейшем в ручном режиме провести его проверку и необходимую корректировку [7].

Тем не менее, далеко не все прикладные модули МИС разумно переносить в режиме «как есть». Основных причин тому две: использование слишком специфичных для Oracle структур данных и методов их обработки, а также не самые удачные архитектурные решения, которые возникали на протяжение более чем 20 лет поступательного развития МИС, когда менялись требования, замещались функциональные компоненты, что привело к накоплению неактуального программного кода. В случае механической миграции такое «наследие» также будет перенесено в целевую систему, и для его актуализации придется затратить определенные усилия.

Нужно отметить, что вопрос о критериях необходимости перепроектирования тех или иных компонент во время миграции оказался достаточно острым, поскольку приходилось выбирать между стратегическими и тактическими целями в условиях значительной неопределенности. Одинаково простыми, но и одинаково неработающими с точки зрения принятия решений являются оба крайних варианта: «всё оставить как есть» или «всё переписать». Так при всей заманчивости идеи оставить всё как было, мы вынужденно «консервируем» все ранее принятые решения, в том числе не самые удачные, до которых в продуктивной среде руки обычно не доходят, в виду несопоставимости ресурсоемкости их изменения и полученного на выходе результата. Все нужные механизмы уже написаны и отлажены, а тот факт, что их можно было бы сделать более простыми и эффективными, при наличии уже реализованной и работающей системы сильным аргументом не является. Другое дело, когда готового результата ещё нет и выполняется перенос софта, который в любом случае придется доводить до работоспособности, тестировать и отлаживать. В этом случае оптимизация становится оправданной, поскольку кроме того, что она позволит сократить дальнейшие затраты по работе с программным кодом, она и на этапе миграции позволит сэкономить ресурсы, исключая перенос избыточного кода.

В качестве ситуаций с выгодным перепроектированием можно рассмотреть те случаи, когда возможно перенести работу с данными на JSON-хранилища, предоставляемые платформой. Так классическая разработка в реляционной СУБД предполагает управление данными через DML-операции Insert/Update/Delete согласно SQL-синтаксису. К тому же, поскольку мы работаем с клиентом через платформу, которая использует методологию NORM [8], заключающуюся в применении

соглашений в формате JSON для обмена данными между клиентом и сервером, то помимо манипуляций с таблицами необходимо выполнять манипуляции с JSON, занимаясь его сериализацией/десериализацией. Если работать с базовыми типами данных, то вне зависимости от оптимизации, программного кода будет достаточно много. Причем весь код по манипулированию данными не будет обусловлен никакой бизнес-логикой, он будет исключительно техническим лишь по той причине, что как схемы данных, так и процедуры работы с ними были спроектированы и реализованы ещё на этапе «толстого» клиента, который работал с таблицами напрямую посредствам SQL, а платформа IPS была вынуждена поддерживать эту технологию для того, чтобы успешно работать на унаследованных базах (в том числе параллельно с «толстым» клиентом). Если же с заменой СУБД необходимость обратной совместимости отпадает, то значительное количество усилий, затрачиваемых на её обеспечение, можно исключить и перейти на работу с хранилищами, которые позволяют работать с данными в стиле NORM. При этом платформа позволяет получать и принимать в хранилища JSON-объекты, а внутри данные уже самостоятельно раскладываются по реляционным таблицам или, наоборот, собираются из реляционных таблиц. В результате на некоторых достаточно простых модулях, в которых нет мощной бизнес-логики, а обеспечивается лишь доставка данных с клиента на сервер и обратно (к таковым относятся, например, большинство справочников, документов, интерфейсы списков различных объектов и пр.), можно получить кратные выигрыши в объеме программного кода и, как следствие, сократить сроки миграции и затраты на последующую поддержку.

Для опытной апробации обеих методик миграции был проведен эксперимент: экспертным путем выбраны две группы подсистем, в одной из них схема данных и бизнес-логика нас в целом устраивала и её было решено автоматически конвертировать с последующей ручной корректировкой (модули, связанные с материальным учетом), другая группа функциональных модулей требовала на наш взгляд оптимизации схемы данных и методов их обработки (модули связанные с медицинскими процессами). Были сформированы одинаковые по количеству участников и квалификации группы разработчиков, которые в течение одинакового промежутка времени занимались реализацией перевода своих подсистем. В обоих случаях пользовательский интерфейс и логика его работы оставались неизменными. Результат выполнения поставленной задачи представлен в таблице 1.

Таблица 1. Результаты эксперимента по оценке двух способов миграции

Компоненты / Методы переноса	Клиентские формы	Модули бизнес-логики	Всего по методу
Автоматическая миграция с сохранением логики	98	365	463
Перепроектирование на технологиях платформы	194	114	308
Всего по компоненте	292	479	
Всего в исходной системе на Oracle	3099	4976	
Процент переведенных модулей	9,42%	9,63%	

Проведенный эксперимент по миграции части модулей МИС не предполагал выполнения определенного объема, он оценивал трудоемкость путем оценки выполненной работы за единицу времени. В результате проекта суммарно был произведен перенос чуть менее 10% основной системы как по модулям пользовательского интерфейса, так и по модулям бизнес-логики системы. Нетрудно заметить, что у обеих команд наблюдается существенный перекос в количестве модулей разных типов: так в процессе перепроектирования удалось реализовать в два раза больше клиентских форм при использовании в среднем 0,6 модуля бизнес-логики на одну форму, тогда как в команде автоматического переноса было реализовано в три раза больше модулей бизнес-логики при использовании в среднем 3,7 модуля на одну клиентскую форму.

Полученный результат показывает, что в тех случаях, когда возможна модернизация схемы данных, её перепроектирование позволяет существенно сократить количество унаследованного программного кода, привязанного к старым данным, возложив рутинные операции по манипуляциями ими на платформу, что и было продемонстрировано, когда удалось отказаться от множества блоков бизнес-логики, обслуживающих модули пользовательского интерфейса. Это и позволило существенно ускорить перенос модулей системы (модуль считается перенесенным, когда он полностью повторяет функции аналогичного модуля в исходной системе). Вместе с тем, нельзя не отметить, что

автоматический перевод существенно ускоряет перенос сложного программного кода, если есть задача сохранить его без изменений.

Также в процессе эксперимента оценивалось, насколько повлияет изменение схемы данных в модифицируемых модулях на те модули, которые переносились без изменения схемы данных. Ключевым тут является учёт связности компонент при выборе метода переноса, поэтому естественно выполнять миграцию тесно взаимодействующих модулей в единой для данной группы методологии. На текущем примере можно констатировать, что связность выбранных модулей оказалась достаточно небольшой, что позволило локализовать изменения, и существенного влияния на общий процесс миграции этот фактор не оказал.

Заключение

Обращаясь к задаче импортозамещения и технологической независимости России в целом, можно отметить, что это сложный процесс, ещё далекий от своего завершения, на этом пути находится ещё масса сложностей, с которыми только предстоит столкнуться и их преодолеть. В похожей ситуации находится и наш проект, также вошедший в процесс трансформации и движущийся по этому непростому пути, который пока ещё не пройден. Однако есть все предпосылки, чтобы успешно завершить преобразование МИС и уже в начале 2023 года выйти на уровень промышленной системы с полностью отечественной инфраструктурой, во многом благодаря рискованному, но показавшему свою эффективность решению развивать собственную программную платформу, которая и обеспечивает текущую трансформацию.

Список литературы

[1] Грант С. К., Копина А. А. *Налоговые стимулы разработки отечественного программного обеспечения* // Налоги.– 2022.– № 4.– с. 29–34. ↑94

[2] Кочуров Е. В. *Конструктивный синтез пользовательских интерфейсов Web-приложений* // Программные системы: теория и приложения.– 2013.– Т. **4**.– № 4(18).– с. 3–25. ↑98

[3] Белышев Д. В., Кочуров Е. В. *Анализ методов хранения данных в современных медицинских информационных системах* // Программные системы: теория и приложения.– 2016.– Т. **7**.– № 2(29).– с. 85–103. ↑98, 102

[4] Белышев Д. В., Кочуров Е. В. *Модульные хранилища данных в платформе "Интерин IPS" для PostgreSQL* // Врач и информационные технологии.– 2021.– № S5.– с. 32–43. ↑98

[5] Elamparithi M., Anuratha V. *Review on database migration strategies, techniques and tools* // World Journal of Computer Application and Technology.– 2015.– Vol. **3**.– No. 3.– pp. 41–48. ↑100

[6] Лашманов А. *Импортозамещение: риски и иллюзии* // Открытые системы. СУБД.– 2015.– № 1.– с. 34–35. ↑100

[7] Курако Е. А., Орлов В. Л. *К вопросу миграции баз данных из среды Oracle в среду Postgresql* // Программная инженерия.– 2022.– Т. **13**.– № 1.– с. 32–40. ↑102

[8] Dombrovskaya H., Novikov B., Bailliekova A. *PostgreSQL Query Optimization: The Ultimate Guide to Building Efficient Queries*, 1st ed..– Berkeley, CA: Apress.– 2021.– ISBN 978–1484268841.– 344 pp. ↑103

Поступила в редакцию 20.10.2022;
одобрена после рецензирования 21.11.2022;
принята к публикации 01.12.2022.

Рекомендовал к публикации *д.ф.-м.н. Н Н. Непейвода*

Информация об авторе:

Дмитрий Владимирович Белышев

кандидат технических наук, заведующий лабораторией Исследовательского центра медицинской информатики Института программных систем имени А. К. Айламазяна РАН. Научные интересы: медицинские информационные системы, образовательные технологии, теория управления

0000-0002-0437-4814

e-mail: *belyshev@interin.ru*

Автор заявляет об отсутствии конфликта интересов.

ISSN 2079-3316 **PROGRAM SYSTEMS: THEORY AND APPLICATIONS** **vol. 13**, No 4(55), pp. 93–109
Research Article INFORMATION TECHNOLOGY IN MEDICINE

UDC 61:004.652
10.25209/2079-3316-2022-13-4-93-109

Import Substitution Experience in the Interin PROMIS Alpha Healthcare Information System

Dmitriy Vladimirovich **Belyshev**
Ailamazyan Program Systems Institute of RAS, Ves'kovo, Russia
✉*belyshev@interin.ru* *(learn more about the author in Russian on p. 107)*

Abstract. Import substitution of database management systems is currently the foremost task facing many information system vendors. In the paper, we consider the experience of migrating the Interin PROMIS HIS from Windows/Oracle technologies to Linux/PostgreSQL based technologies recommended by the Register of Russian Software. The paper also provides an assessment of two approaches to migration of information system modules, namely redesign using the tools of the Interin Alpha PG platform and automatic conversion using in-house tools. (*In Russian*).

Key words and phrases: healthcare information system, import substitution, PostgreSQL

2020 *Mathematics Subject Classification*: 68P15; 68P20, 92C50

For citation: Dmitriy V. Belyshev. *Import Substitution Experience in the Interin PROMIS Alpha Healthcare Information System* // Program Systems: Theory and Applications, 2022, **13**:4(55), pp. 93–109. (*In Russian*). `http://psta.psiras.ru/read/psta2022_4_93-109.pdf`

References

[1] S. K. Grant, A. A. Kopina. "Tax incentives to the development of domestic software", *Nalogi*, 2022, no. 4, pp. 29–34 (in Russian). ↑94

[2] Ye. V. Kochurov. "Constructive synthesis of Web-based application user interfaces", *Program Systems: Theory and Applications*, **4**:4(18) (2013), pp. 3–25 (in Russian). ↑98

[3] D. V. Belyshev, Ye. V. Kochurov. "Analysis of data storage methods for modern healthcare information systems", *Program Systems: Theory and Applications*, **7**:2(29) (2016), pp. 85–103 (in Russian). ↑98, 102

[4] D. V. Belyshev, Ye. V. Kochurov. "Modular data warehouses in the Interin IPS platform for PostgreSQL", *Vrach i informatsionnyye tekhnologii*, 2021, no. S5, pp. 32–43 (in Russian). doi URL ↑98

[5] M. Elamparithi, V. Anuratha. "Review on database migration strategies, techniques and tools", *World Journal of Computer Application and Technology*, **3**:3 (2015), pp. 41–48. doi URL ↑100

[6] A. Lashmanov. "Import substitution: risks and illusions", *Otkrytyye sistemy. SUBD*, 2015, no. 1, pp. 34–35 (in Russian). URL ↑100

[7] Ye. A. Kurako, V. L. Orlov. "On the issue of migrating databases from Oracle to Postgresql", *Programmnaya inzheneriya*, **13**:1 (2022), pp. 32–40 (in Russian). doi ↑102

[8] H. Dombrovskaya, B. Novikov, A. Bailliekova. *PostgreSQL Query Optimization: The Ultimate Guide to Building Efficient Queries*, 1st ed., Apress, Berkeley, CA, 2021, ISBN 978–1484268841, 344 pp. ↑103