

## РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ РАБОТЫ С КОРПОРАТИВНЫМИ ИНФОРМАЦИОННЫМИ СИСТЕМАМИ

Ю. Г. Платонов

Институт вычислительных технологий СО РАН, 630090, Новосибирск, Россия

---

УДК 004.738.5:004.942

Предложена универсальная методика, обеспечивающая работу пользователей произвольной Enterprise-системы, использующих в качестве рабочей станции любое мобильное устройство или иное устройство с ограниченными ресурсами. Архитектура системы построена на основе шаблона CQRS. Применение описанного метода возможно без изменения серверной части приложения. Для эффективной работы не требуется стабильное соединение с Интернетом, клиентская часть приложения эпизодически устанавливает связь с сервером для синхронизации данных. Описаны технология авторизации удаленных пользователей и решение проблемы безопасности данных при удаленной работе.

**Ключевые слова:** CQRS, command and query responsibility segregation, мобильные устройства, распределенные системы, корпоративные системы, синхронизация данных.

In the article the author describes the new universal method to provide the mobile work of a user of any Enterprise system. Author uses the CQRS Template as the basis of the solution, and justifies the appropriateness of its usage for any device with the limited resources. The method requires no changes for a server part of the system and doesn't require the stable internet connection. When the mobile client part of the application connects to the server from time to time for data synchronization, it's enough for the successful work. Also the article describes the method to authorize the remote users and provide the data security for the remote work.

**Key words:** CQRS, Command and Query Responsibility Segregation, mobile devices, distributed systems, Enterprise systems, data synchronization.

**Введение.** В настоящее время широкое распространение получили мобильные устройства, такие как коммуникаторы и планшетные компьютеры, использующие различные операционные системы. Вследствие доступности Интернета большинство общеизвестных программ используются не только на стационарных компьютерах, но и на мобильных устройствах. В то же время постоянно разрабатываются приложения для мобильных устройств. Так, в связи с распространением iPad появились новые программы для MacOS.

Рынок приложений для мобильных устройств представляется насыщенным только на первый взгляд, поскольку большинство этих приложений являются либо офисными (например, пакеты OpenOffice или Microsoft Office), либо мультимедийными и коммуникативными (например, Skype), в то время как пользователям мобильных устройств удобно использовать корпоративные приложения, например складские и бухгалтерские программы. В настоящее время такая возможность практически отсутствует (исключением являются некоторые корпоративные приложения, имеющие web-интерфейс), что обусловлено следующими причинами:

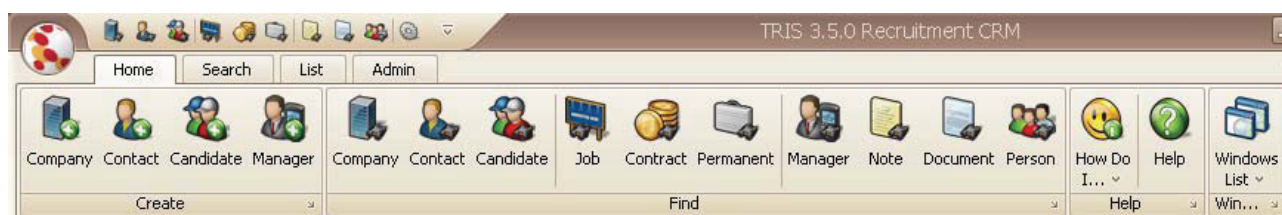


Рис. 1. Меню с основными функциями системы TRIS, доступное на стационарном компьютере

— корпоративные приложения предполагают использование сложного функционала клиентской части, для отображения которого необходимы большие вычислительные ресурсы, недоступные для мобильных устройств;

— доступ к данным корпоративной базы данных (БД) может потребоваться одновременно нескольким пользователям, причем одни из них могут обращаться к ней со стационарных компьютеров, другие — с удаленных мобильных устройств;

— нередко возникает необходимость продолжения работы удаленного пользователя даже при временном отсутствии доступа к Интернету;

— мобильное приложение, поддерживающее работу с удаленными пользователями, должно обеспечивать защиту данных от несанкционированного доступа, т. е. обеспечивать надежные механизмы однозначной идентификации пользователей и защиты передаваемых данных.

Таким образом, полноценная работа удаленного мобильного пользователя корпоративной информационной системы требует решения ряда дополнительных задач. В настоящее время этот слабо разработанный сегмент рынка является перспективным, поскольку любая корпоративная система с возможностью удаленной работы с мобильного устройства более конкурентоспособна.

Предложенная методика применяется компанией Recruitment Systems Pty Ltd (Австралия), в которой мобильное приложение проходит подготовку к промышленной эксплуатации. Компания занимается разработкой, внедрением и сопровождением программного обеспечения для рекрутингового бизнеса. Основным программным продуктом компании Recruitment Systems является информационная система TRIS, в настоящее время идет процесс переноса этой системы на мобильные носители.

Для того чтобы оценить преимущества использования приложений на мобильном устройстве по сравнению со стационарным компьютером, проанализируем многолетний опыт компании Recruitment Systems. Система TRIS — основной продукт компании — представляет собой многофункциональный информационный комплекс, разработанный для нужд агентств по трудоустройству [1] (рис. 1).

Любой сотрудник кадрового агентства делает большое количество звонков, регулярно проводит собеседования с соискателями вне офиса, и ему постоянно необходима актуальная информация об имеющихся вакансиях. Поэтому создание мобильного приложения, совместимого с системой TRIS и дающего возможность любому зарегистрированному пользователю получать и редактировать информацию из БД, используя любое удаленное мобильное устройство, значительно повысит конкурентоспособность системы.

Суть предлагаемого решения заключается в перемещении клиентской части корпоративной программы на мобильное устройство, при этом серверная часть приложения будет по-прежнему храниться на стационарном компьютере. Недостатком предлагаемого решения является невозможность получения актуальных данных при отсутствии доступа к Интер-

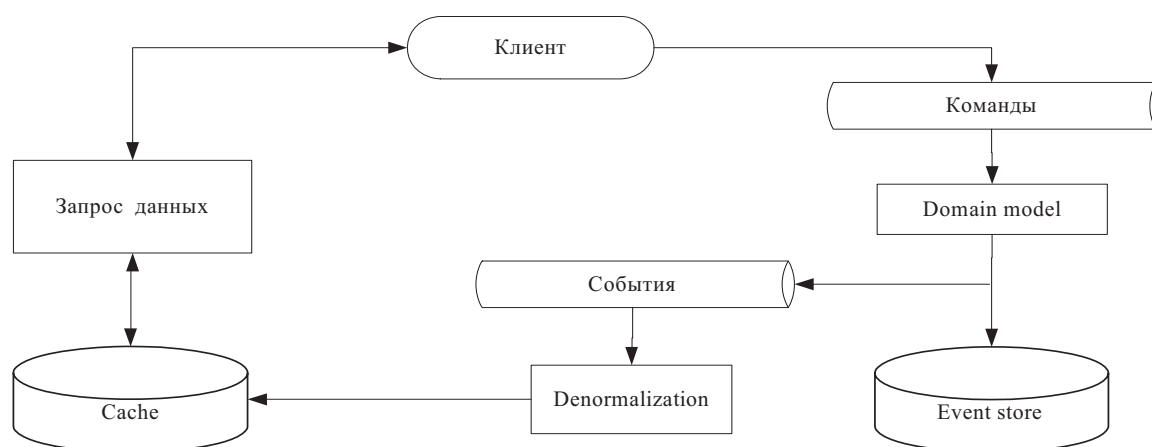


Рис. 2. Схема работы шаблона CQRS

нету. В этом случае пользователь вынужден работать с данными, полученными во время последней синхронизации с главной БД.

Если эффективность деятельности удаленного специалиста зависит, например, от наличия товара на складе, то переноса клиентской части корпоративной программы складского учета может оказаться недостаточно, поскольку невозможно гарантировать доступность корпоративного сервера. В этом случае специалисту потребуется самостоятельно учитывать предполагаемое изменение ассортимента товара за время отсутствия информации от сервера либо изменять бизнес-процесс. Однако даже в этом случае перенос клиентской части приложения на мобильное устройство увеличит эффективность деятельности предприятия, так как ранее либо удаленные сотрудники были вынуждены прогнозировать наличие товара на складе, либо складские запасы были достаточно велики.

Перенос клиентской части корпоративного приложения на мобильное устройство осложняется ограниченностью вычислительных ресурсов и возможной нестабильностью или низкой производительностью средств беспроводной связи при удалении от источника сигнала. Решить проблему нестабильности связи можно путем понижения трафика между устройствами или создания дополнительного хранилища информации непосредственно на мобильном устройстве. Для этих целей предлагается использовать шаблон Command and query responsibility segregation (CQRS).

**1. Краткое описание CQRS.** Шаблон CQRS является продолжением идеологии Command and query separation (CQS), предложенной Б. Мейером [2]. Идея Б. Мейера была перенесена на архитектурный уровень Г. Юнгом (официальный блог Г. Юнга <http://codebetter.com/gregyoung>).

Схема работы шаблона CQRS приведена на рис. 2 (Cache — база данных, представленная в терминах клиентской части приложения и адаптированная для чтения; Domain model (модель области определения) — “сеть взаимосвязанных объектов, в которой каждый объект представляет собой отдельную значащую сущность, может быть, настолько большую, как корпорация, или настолько малую, как строка из формы заказа” [3]; Denormalization — процесс синхронизации Cache и Event store путем интерпретации опубликованных событий сервера как изменения Cache; Event store — база данных, адаптированная для записи информации в формате сервера).

Концепция шаблона состоит в разделении всех внешних действий клиентского приложения на запросы к данным (query) и команды на изменение данных (command). Для проверки корректности данных и оценки возможности исполнения команд в шаблоне используется

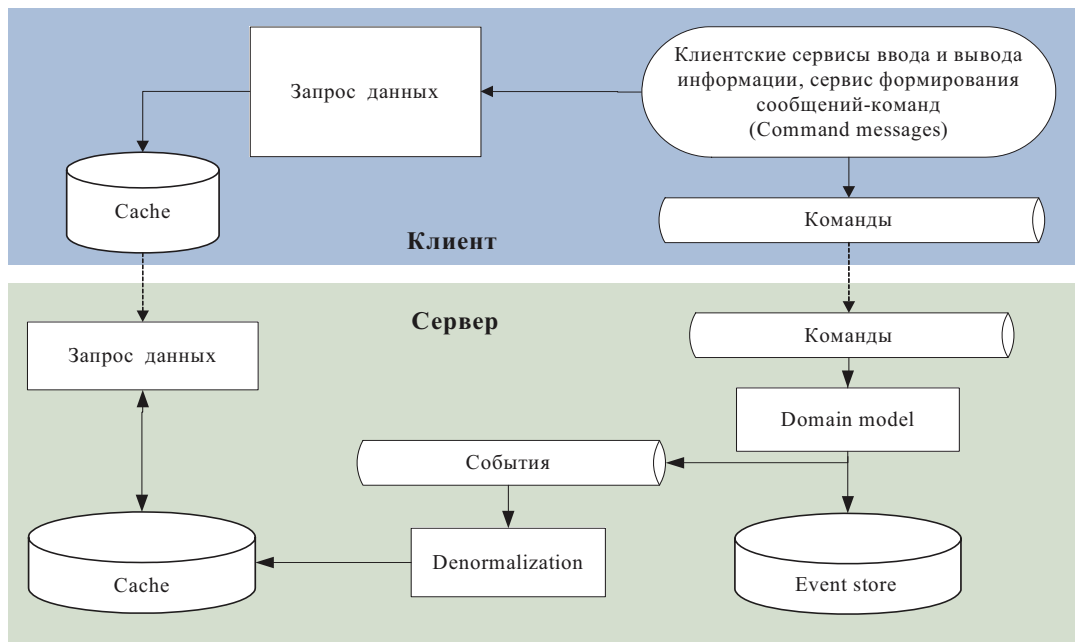


Рис. 3. Схема взаимодействия измененного клиента с сервером, реализованного с использованием шаблона CQRS

модель области определения. Все коммуникации осуществляются через систему сообщений, имеющих определенную структуру (contract). Таким образом, весь процесс взаимодействия клиентской и серверной частей приложения укладывается в две цепочки событий.

Для получения набора данных клиент отправляет публичной части сервера запрос, на основании которого формируется выборка объектов из Cache, описанных в терминах клиента, после чего она возвращается клиенту как результат работы сервера.

Для создания, удаления или изменения существующего объекта клиент отправляет на сервер запрос-команду. Запрос проверяется на корректность и возможность исполнения и ставится в очередь. При достижении вершины очереди команда передается в сервис, в результате работы которого формируется публичное сообщение, сохраняемое сначала в Event store, а после процедуры денормализации — в Cache. Подобное разделение позволяет избежать лишних преобразований и снизить сетевой трафик.

**2. Технология организации удаленной работы с Enterprise-приложениями.** На рис. 2 видно, что клиентская часть любого приложения представляет собой “черный ящик”, посылающий различные распоряжения серверу. В условиях ограниченных вычислительных ресурсов клиента это может оказаться серьезной проблемой.

На рис. 3 приведена та же схема, адаптированная для работы с мобильным приложением. В результате переноса на мобильное устройство изменились клиентская часть приложения и его взаимодействие с сервером, а серверная часть осталась неизменной. Из рис. 3 следует, что при создании подобных систем особое внимание должно уделяться системе взаимодействия сервера и клиента.

Для стабильной работы клиентской части предлагается использовать на ее стороне дополнительный набор данных, сохраненных в терминах клиентской части, режим чтения которого активизируется только в случае автономной работы клиентского приложения.

В случае нормальной связи серверной части хранилище данных на мобильном устройстве, предназначенное для работы с клиентским приложением, производит только запись объектов по принципу стека (см. п. 5).

Если связь с сервером нарушена, пользователь использует последнюю версию полученных данных. Это может быть полная копия БД, если отведенные для этого ресурсы мобильного устройства позволяют сохранить такое количество данных, либо ее часть (например, 500 объектов БД, использованных последними). В этом случае хранилище данных на мобильном устройстве временно используется как роль БД (см. п. 6).

Независимо от стабильности связи с сервером все поступающие от клиентской части команды на добавление новых объектов, изменение и удаление существующих объектов сохраняются в дополнительной очереди, также размещенной на стороне клиентской части. Когда появляется возможность передачи этих команд на сервер (при наличии связи с сервером — в тот же момент, при отсутствии — по мере восстановления соединения), происходит дополнительная проверка актуальности выполненных действий, и, в случае если проверка выполнена успешно, осуществляются дальнейшая передача команд на сервер и исполнение их в штатном порядке. В этом случае клиентскую часть приложения можно представить как объединение двух сервисов: 1) стандартной программы клиента, обеспечивающей взаимодействие с пользователем через устройства ввода-вывода и некоего хранилища последних использованных данных; 2) очереди команд, сохраненных клиентской частью и поставленных в очередь для исполнения на сервере либо при активизации сервера, либо при наличии свободного места в очереди команд на сервере. Далее приведены основные аспекты работы подобной схемы (см. пп. 3, 7).

Следует отметить, что любая клиентская часть Enterprise-системы для мобильных устройств должна иметь собственный оригинальный графический пользовательский интерфейс, отличный от интерфейса приложения для стационарного компьютера (см. п. 3).

**3. Решение проблемы недостаточности вычислительных ресурсов и объема памяти.** Поскольку предложенный метод использует файловое пространство мобильного клиентского приложения для хранения информации об объектах, может возникнуть проблема недостаточности объема памяти. Клиентская часть может иметь сложный функционал отображения данных, кроме того, сам объект данных может быть сложным и обладать большим количеством связей с другими объектами системы.

Проблему недостатка файлового пространства для хранения предлагается решить путем дополнительной настройки клиентского приложения, позволяющей пользователю определить принцип размещения объектов (например, отказаться от сохранения объектов одного типа ради сохранения всех объектов другого типа).

Прежде чем приступать к практическому решению проблемы недостаточности вычислительных ресурсов, необходимо определить круг задач, которые будет решать данное приложение. Например, если мобильное приложение должно решать задачи оперативного управления, то основные имеющиеся вычислительные ресурсы следует направить на отображение объектов. Если основная задача приложения — получение данных, формирование итоговых данных, полученных расчетным путем, и передача их третьему приложению, то задача заключается в максимальном упрощении пользовательского интерфейса с целью наилучшего выполнения процесса вычисления. Такая определенность целесообразна при проектировании области определения объектов для мобильного клиентского приложения, а также для упрощения переработки пользовательского интерфейса с учетом специфики мобильных устройств, которые, как правило, поддерживают мануальное управление (без использования манипулятора “мышь”) и имеют нестандартный экран небольших размеров с невысоким разрешением.

Далее, все объекты области определения необходимо разделить на те, к которым пользователь мобильного клиентского приложения сможет получить доступ, и те, которые будут ему недоступны, а затем проверить все связи между доступными объектами. Если связь между объектами не несет информации для мобильного приложения, то объекты не должны быть связаны в терминах мобильного клиента (рис. 4).

Рис. 4. Система TRIS на стационарном компьютере, отображающая большее количество данных, чем мобильный вариант этого приложения

в двух аспектах: 1) общая защита от несанкционированного доступа к информации; 2) различные режимы аутентификации пользователя в зависимости от доступности центра или сервиса авторизации. Рассмотрим оба аспекта.

Если центр или сервис авторизации доступен, то политика авторизации удаленного пользователя не будет отличаться от политики, принятой для локального клиентского приложения. Исключением могут служить корпоративные приложения, использующие для авторизации ключи электронной цифровой подписи [4]. В зависимости от степени защиты самих сообщений обеспечивается соответствующий уровень сложности шифрования, определяемый общей политикой доступа к информации приложения.

Если мобильный клиент работает в режиме, когда сервер недоступен, предлагается предоставлять доступ только пользователям, ранее авторизованным сервисом. Для этого предлагается на стороне мобильного клиента иметь список подобных пользователей с минимальным способом авторизации (например, использовать способ защиты информации паролем). Список может быть зашифрован тем же способом, что и данные, сохраненные на стороне клиента. При таком способе идентификации пользователя защита приложения не будет нарушена, так как мобильный клиент может быть авторизован только пользователем, ранее уже одобренным сервисом авторизации. Пользователю, авторизованному подобным образом, будет доступна только та часть базы данных, доступ к которой был предоставлен ему ранее при авторизации штатным способом. В случае если сервис авторизации вновь станет доступным, пользователь должен будет авторизовываться штатным способом. В случае отказа в авторизации в период недоступности сервиса пользователь не сможет пройти штатную авторизацию и его дальнейшая работа в системе будет невозможна ни в каких режимах системы (рис. 5).

Избавление от подобных фантомных связей позволяет существенно снизить трафик между мобильным клиентом и сервером, уменьшить размер сохраняемых объектов и как следствие уменьшить объем памяти, необходимой для хранения объектов на жестком диске мобильного устройства, упростить функционал данного приложения (при условии, что сама задача, за выполнение которой отвечает приложение, остается работоспособной), а также пользовательский интерфейс.

Заметим, что, если мобильному приложению требуется информация о связанном объекте, ее можно запросить у сервера с помощью дополнительного запроса.

#### 4. Авторизация и безопасность передачи данных.

Процесс авторизации для распределенных систем следует рассматри-

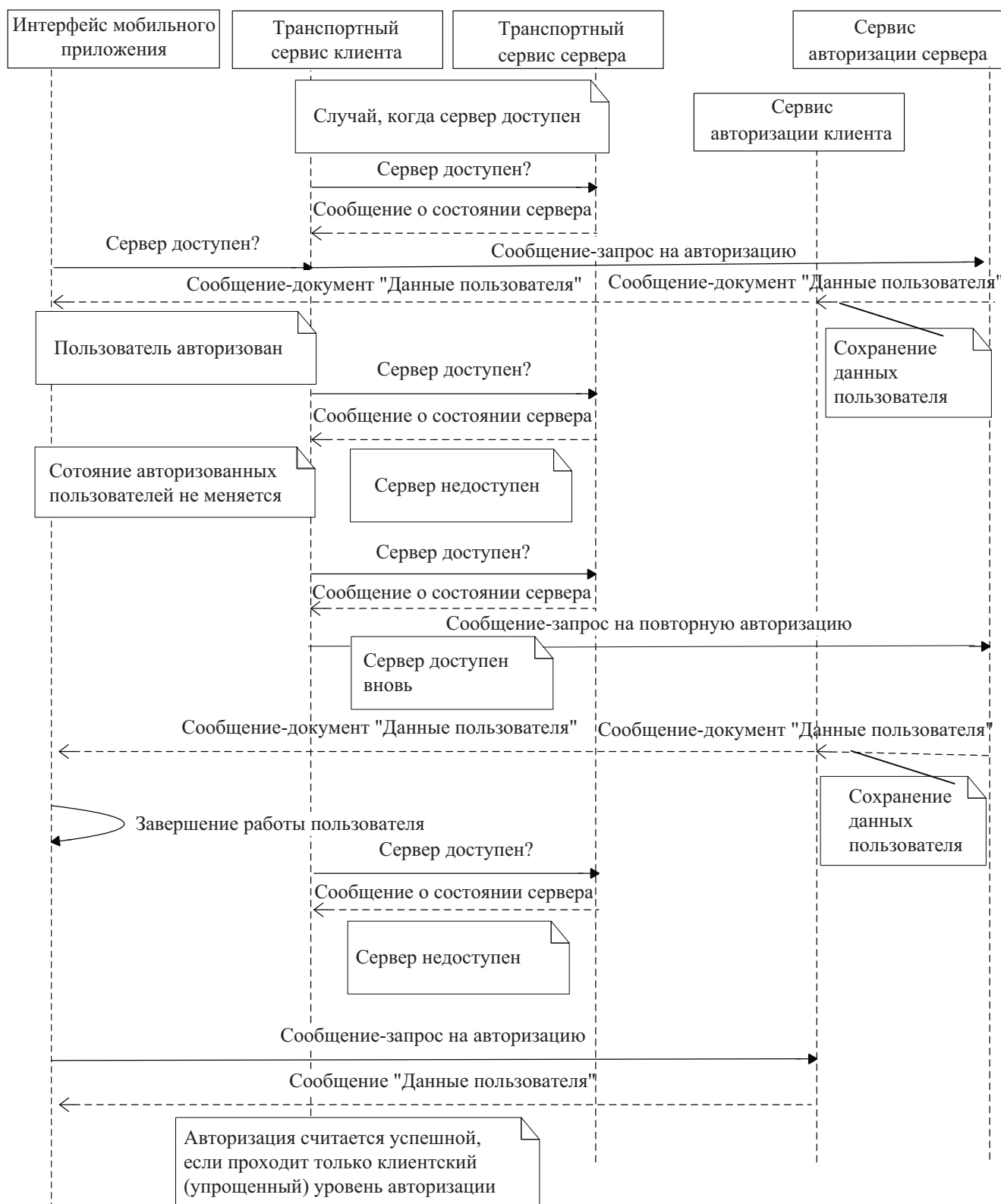


Рис. 5. Диаграмма последовательностей авторизации пользователя

**5. Штатная работа системы.** Рассмотрим основные ситуации, возникающие при работе мобильного пользователя в условиях, когда удаленный сервер корпоративного прило-

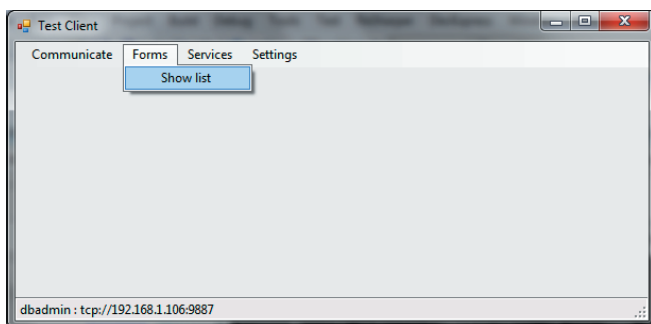


Рис. 6. Главное окно клиентской части приложения TRIS, разработанного для мобильных устройств

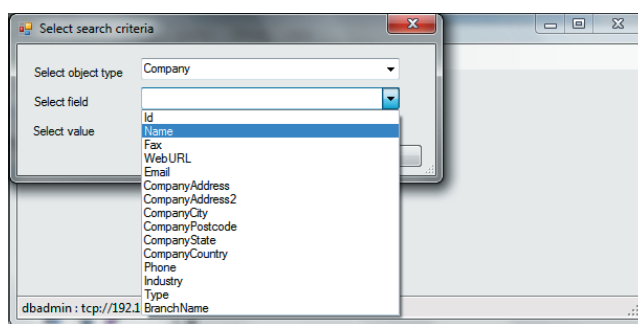


Рис. 7. Мобильный TRIS. Доступными являются только критерии поиска объектов, актуальные для мобильного приложения

жения доступен для взаимодействия с мобильным клиентским приложением. Такой режим работы будем называть штатным.

Авторизованный пользователь запрашивает данные определенного типа по определенному критерию (рис. 6–9).

Клиентское приложение публикует сообщение-событие (Event message), которое доставляется серверу. В ответ сервер публикует сообщение (Document message) с набором документов, удовлетворяющих запросу (рис. 10).

Опубликованное сообщение отображается на устройстве вывода клиентского приложения и одновременно заполняет стек объектов данного типа клиентского приложения (рис. 11). В то же время мобильный пользователь получает доступ к запрашиваемым данным.

Следует отметить, что пользователь может регулировать размер стека, используя сервис настроек, в зависимости от наличия свободного места на устройстве и частоты автономной работы (см. п. 4).

Если размер стека достаточно велик, пользователь сможет просматривать и редактировать такое же количество объектов, как и при работе со стационарной версией приложения.

После изменения, удаления или добавления объекта клиентское приложение публикует сообщение-команду (Command message), которая размещается в очереди команд на мобильной клиентской части приложения и по мере обмена информацией с сервером передается на шину команд сервера, а затем либо исполняется (Domain model)-сервисом, либо отвергается как некорректная или неактуальная.

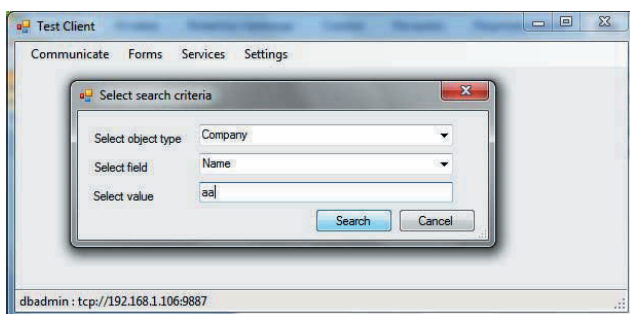


Рис. 8. Мобильный TRIS. Выбор критерия для поиска объекта в центральной БД

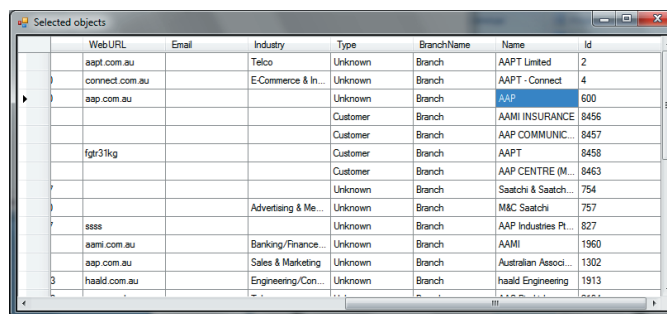


Рис. 9. Мобильный TRIS. Список объектов, полученных в ответ на запрос мобильного пользователя

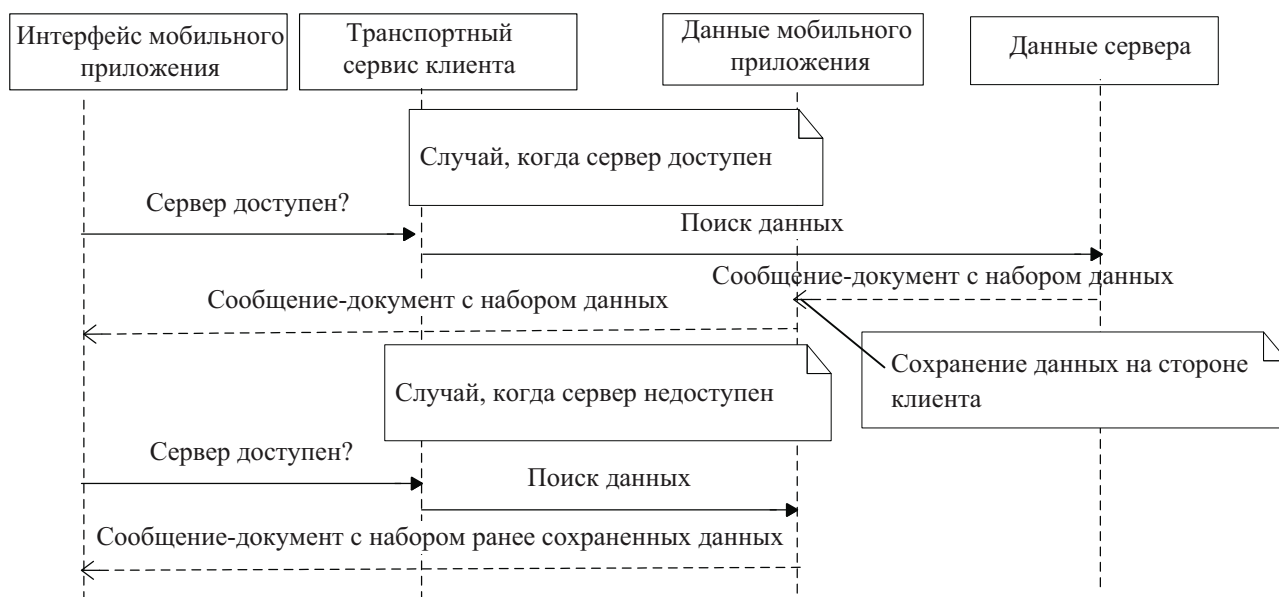


Рис. 10. Последовательность получения данных клиентской частью приложения, установленной на мобильном устройстве, при работе системы в штатных условиях

В случае неуспешного исполнения команды мобильное клиентское приложение получает извещение о результате. Аналогично, в случае если команда выполнена успешно, пользователь получает соответствующее уведомление.

При использовании предложенной схемы повышается надежность всей системы в случае неожиданного нарушения соединения с сервером.

**6. Работа в условиях недоступности сервера.** Исходя из особенностей функционирования мобильных устройств необходимо учитывать возможность работы с корпоративным приложением в условиях временного отсутствия доступа к Интернету. В качестве решения данной проблемы можно предложить авторизованный сеанс работы мобильного пользователя с набором объектов, полученных в результате сохранения предыдущих сеансов в режиме, когда стационарный сервер был доступен.

При организации работы рассматриваемой распределенной системы в случае недоступности сервера могут возникнуть проблемы авторизации, организации работы с ранее сохраненными данными, отложенного сохранения.

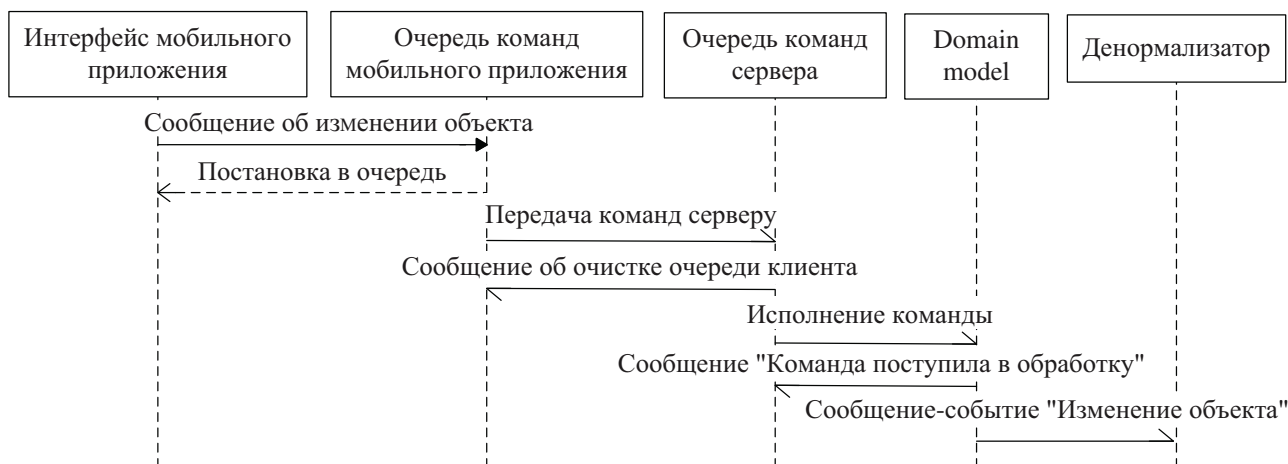


Рис. 11. Последовательность обработки команд при работе системы в штатных условиях

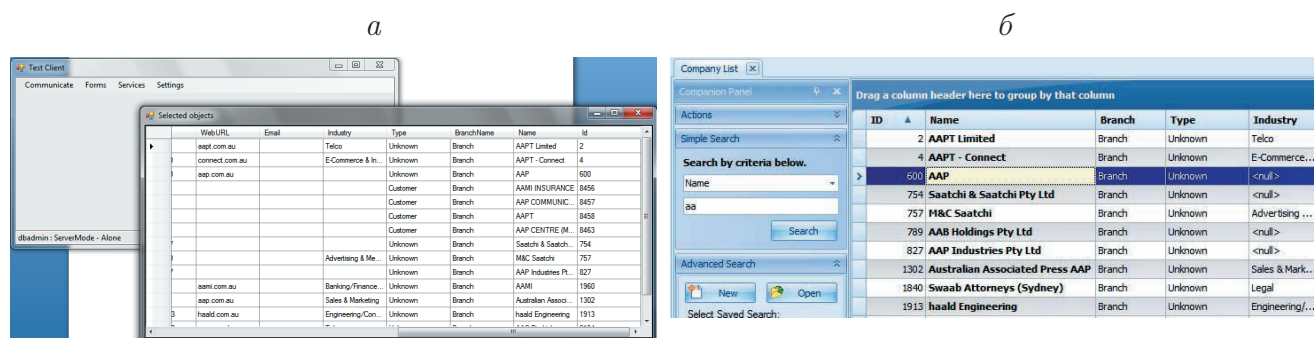


Рис. 12. Отображение объектов, полученных в ответ на запрос пользователя:

*a* — мобильный TRIS; *б* — стационарный TRIS

Предложенная выше методика получения, обработки и хранения данных мобильным приложением позволяет авторизованному пользователю, как и в штатной ситуации, запрашивать данные определенного типа по какому-либо критерию выбора (рис. 12). На рис. 12 видно, что клиентская часть приложения отображает тот же запрашиваемый список объектов, что и при работе в штатном режиме, поскольку все объекты были ранее загружены в хранилище данных (стек). После получения данных на предоставление система публикует сообщение-событие (Event message).

Клиентское приложение создает сообщение-команду, которая в зависимости от ситуации либо изменит объект, либо создаст новый, либо удалит объект из стека и сохранится в очереди команд (рис. 13). В случае повторного редактирования объекта пользователем сообщения-команды сохраняются в очереди последовательно, а в случае удаления уже отредактированного объекта сохраняется только сообщение-команда на удаление.

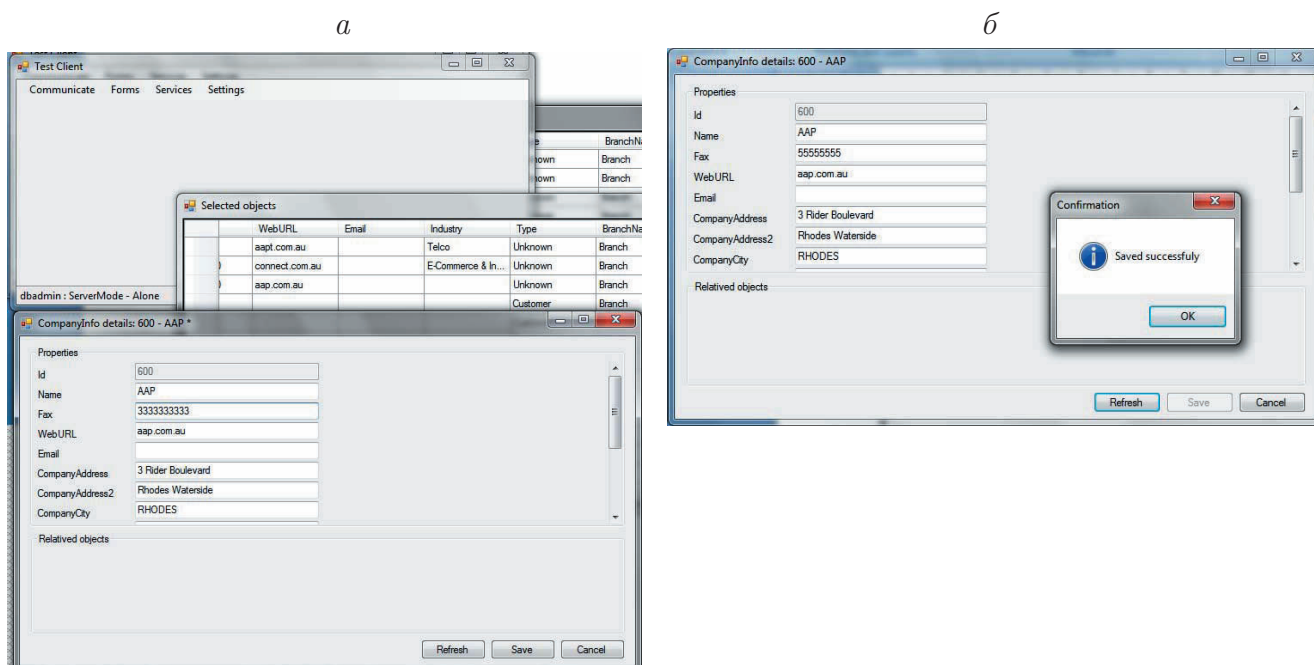


Рис. 13. Мобильный TRIS:

*a* — редактирование объекта на мобильном устройстве в автономном режиме (ServerMode — Alone); *б* — информация об успешном сохранении данных



Рис. 14. Последовательность обработки команд в условиях недоступности сервера

Следует отметить, что при подобной авторизации на клиентском приложении должна существовать служба, которая через определенные промежутки времени будет проверять возможность соединения с сервером (рис. 14). В случае восстановления доступа к серверу клиентское приложение должно асинхронно выполнить следующую последовательность операций.

1. Штатная авторизация пользователя. Если авторизация текущего пользователя невозможна вследствие того, что информация о нем была удалена на сервере либо ему был запрещен доступ в систему, то пользователь будет удален из списка ранее авторизованных пользователей, будет очищена очередь команд и клиентское приложение перейдет в неавторизованный режим работы.

2. На сервере выполнена вся очередь команд, сохраненных клиентским приложением. В случае, когда сервер не может выполнить команду, пользователь должен быть извещен посредством специального сообщения-события.

3. Обновлено объекты, находящиеся в Cache клиентского приложения.

Еще одной важной задачей является отложенное сохранение данных в случаях, когда пользователь добавил, изменил или удалил данные.

Если в штатном режиме работы системы проблему конкурентного сохранения данных можно решить стандартными методами [5], то в случае автономной работы мобильного клиентского приложения эти методы не смогут обеспечить эффективную работу системы, так как время существования измененных данных может быть достаточно большим.

Данную ситуацию целесообразно представить в виде взаимодействия двух независимых баз данных (стационарная база данных на сервере и данные в Cache клиентского приложения), причем технология этого взаимодействия должна обеспечить их объединение.

В качестве критерия оценки актуальности информации можно выбрать время последнего обновления (с учетом миллисекунд) и часового пояса сервера. Тогда весь процесс сохранения измененных данных можно разделить на ряд уникальных процессов и описать их по отдельности.

Более простой случай автономной работы мобильного пользователя — неконкурентные процессы, представляющие собой процессы синхронизации данных, при выполнении которых не возникает коллизий данных. Например, изначально на сервере существовал объект, который мобильное приложение сохранило в Cache, через некоторое время оно сформировало серверу сообщение об удалении этого объекта, при этом другие пользователи изменений данного объекта не производили.

При сохранении пользователем измененных данных клиентская часть приложения информирует его о том, что выполненные им изменения были успешно сохранены в очереди команд.

Процессы добавления данных целесообразно разделить на добавление сущностей (бизнес-объектов, определяющих информационную систему) и пополнение справочников (второстепенных наборов данных, связанных с сущностями). Если изменение содержимого справочника происходит нечасто, то справочник информационной системы можно рассматривать в рамках работы мобильного приложения как статический объект.

Процесс добавления объектов в информационную систему при автономной работе несложен. Единственной проблемой, возникающей при добавлении объектов, является создание объектов-дубликатов. Корректное решение данной проблемы предполагает, что при восстановлении связи с сервером и сохранении в центральной базе данных нового объекта система автоматически выполняет проверку присутствия в системе объекта с такими же данными, после чего список подобных объектов (предполагаемых дубликатов) предоставляется пользователю. Затем пользователь самостоятельно определяет, следует ли добавлять в систему новый объект. Таким образом, предлагается следующий алгоритм решения проблемы объектов-дубликатов:

1. В процессе добавления объекта формируется запрос существования объектов в базе данных по набору критериев, заранее определенных в модели для данной сущности. В отсутствие ранее созданных подобных объектов процесс добавляет объект в систему, в противном случае мобильному пользователю выводятся существующие варианты объектов.

2. Пользователь решает, добавлять ли новый объект в систему или использовать уже существующий объект. В случае выбора существующего объекта предложенный к добавлению объект заменяется уже существующим, и все его данные обновляются.

Нередко данная задача оказывается сложной даже для опытного пользователя, однако предлагаемый подход помогает эффективно избежать появления в системе одинаковых записей.

Рассмотрим процесс сохранения данных отредактированного объекта. Как и в случае удаления объектов, важно определить, какие данные являются актуальными. Для этого используется тот же критерий — дата последнего обновления. В случае если сохраняемый объект на сервере не менялся, процесс будет неконкурентным и сервер лишь обновит поля сохраняемого объекта.

В случае изменения данного объекта и на мобильном клиенте, и на сервере сервер должен передать пользователю список конфликтов, что позволит скорректировать объект и сохранить финальные изменения. Следует отметить, что при рассмотрении подобного процесса у проектировщика системы имеется дополнительная возможность избегать конфликтных ситуаций, возникающих в момент сохранения объекта. Рассмотрим следующую ситуацию: пользователь мобильного приложения получил объект данных на редактирование, после чего эти данные были изменены как на сервере (другим пользователем), так и на мобильном устройстве. При обычном подходе, если мобильный пользователь попытается сохранить

полученный объект, то в ответ получит набор конфликтных записей и будет вынужден редактировать объект еще раз. При этом число подобных итераций может быть достаточно большим.

Для оптимизации и уменьшения времени, затрачиваемого на сохранение объектов, можно предложить рассматривать процесс сохранения объекта как процесс сохранения списка измененных полей в объекте, заранее определенном по какому-либо первичному ключу. В этом случае для определения критерия актуальности значения в поле можно предложить значение, которое было в поле до его редактирования. Тогда этот критерий совпадет со значением в поле объекта, находящегося на сервере, и можно предположить, что данное поле не редактировалось или редактировалось таким образом, что его последняя редакция совпадает с начальным значением. Такое поле не может считаться конфликтным, несмотря на то что при традиционном подходе ситуация с кратным редактированием попадает в список конфликтов. Еще одним приемом оптимизации процесса сохранения объекта следует считать дополнительную обработку очереди команд в ситуации, когда мобильный пользователь дважды отредактировал один и тот же объект.

**7. Возможность применения метода для стандартных корпоративных приложений.** Использование архитектурного шаблона CQRS позволяет внести изменения в клиентскую часть практически любого корпоративного приложения и перенести его на мобильную платформу (см. п. 2). Однако в настоящее время большинство корпоративных приложений используют стандартную трехуровневую архитектуру [6]. При этом зачастую к началу процесса проектирования уже имеется действующее корпоративное решение со сложной структурой и заказчик не готов к большим капиталовложениям для его перевода с одной архитектурной платформы на другую. Предлагаемый метод может быть использован также в ситуации, когда разработчик должен сохранять начальное трехуровневое решение.

В результате анализа современного рынка корпоративных информационных систем с точки зрения закрытости программного кода и серверного функционала все существующие системы можно разделить на три основных типа по сложности интеграции с мобильным клиентским приложением. К корпоративным приложениям первого типа следует отнести информационные системы, имеющие настраиваемые коммуникации с внешними приложениями. Наиболее характерный пример отечественного рынка корпоративных приложений — программные продукты компании “1С”. В новой версии платформы компания “1С” разработала специальную компоненту для внешних соединений, работающую по принципу Com-объекта как на стороне “1С: Предприятия”, так и на стороне внешнего приложения (официальный сайт компании “1С” — интеграция с внешними приложениями [http://v8.1c.ru/overview/Term\\_000000581.htm#1](http://v8.1c.ru/overview/Term_000000581.htm#1)). На рис. 15 приведена схема подобного взаимодействия.

В качестве других примеров информационных систем можно рассмотреть несколько приложений, объединенных BizTalk Server (официальный сайт продукта Microsoft BizTalk Server

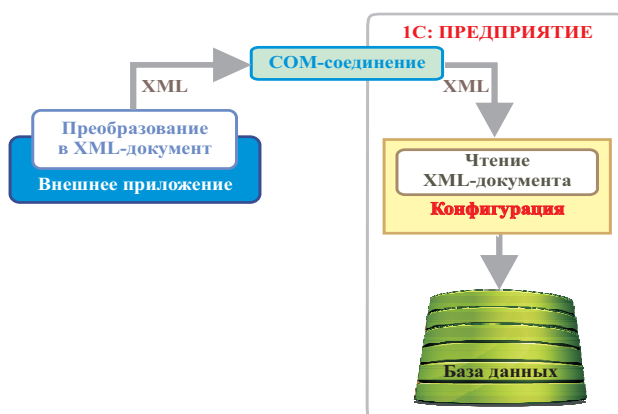


Рис. 15. Схема взаимодействия внешних приложений с информационной системой “1С: Предприятие”

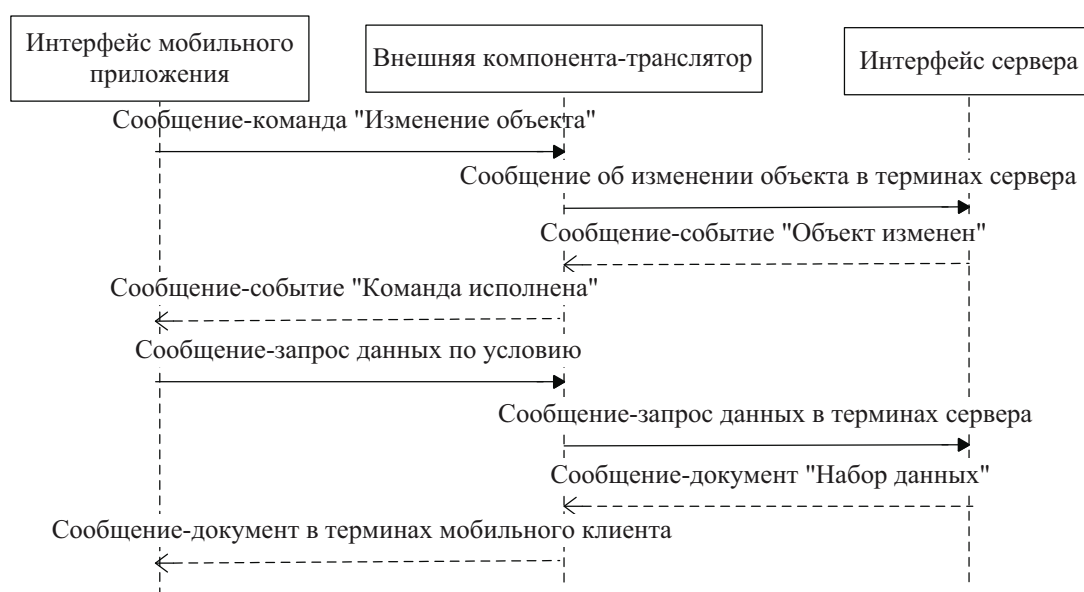


Рис. 16. Последовательность работы промежуточной внешней компоненты-транслятора

<http://www.microsoft.com/biztalk/en/us/overview.aspx>), в данном случае действующем как центр интеграции. Он преобразует сообщение во внутреннее представление и посылает на вход преобразователя, который трансформирует сообщение в выходной XML-формат по правилам, заданным в описании преобразования. Далее с помощью выходного преобразователя этот XML-файл преобразуется в один из поддерживаемых форматов (XML, EDI и Flat file) и отправляется по одному из транспортных протоколов приложению-получателю.

В случае создания мобильной клиентской части для подобных систем предлагается расширить методику, предложенную в п. 2, добавив дополнительную промежуточную компоненту между мобильным клиентом и трехуровневым сервером корпоративного приложения (рис. 16). Из рис. 16 следует, что эта компонента, являющаяся простым синхронным транслятором, трансформирует сообщения-запросы, описанные в терминах клиента, во внутреннее представление сервера и публикует их в заранее известный канал коммуникации. Аналогично она поступает с сообщениями, публикуемыми сервером.

К корпоративным приложениям второго типа следует отнести приложения, имеющие в целом закрытый программный код, не имеющие интерфейса для коммуникаций с внешними информационными системами, но имеющие некоторый API (application public interface) для доступа внешних компонент. Необходимо отметить, что на современном рынке существует большое количество приложений этого типа, характеризующихся наличием дополнительной компоненты (например, использующей SOAP протокола (официальный сайт спецификации SOAP <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/>) или библиотеки, имеющей широкий набор параметров для запуска с командной строки. Решением, обеспечивающим работу мобильной клиентской части информационной системы для корпоративных приложений этого типа, как и для приложений первого типа, будет создание некоторой промежуточной компоненты-транслятора. Особенность указанной компоненты состоит в том, что набор внешних функций для корпоративных приложений второго типа не подразумевает обратной связи, что не гарантирует исполнение команд мобильного клиента. Поэтому данный функционал должен быть перенесен в промежуточный транслятор (рис. 17).

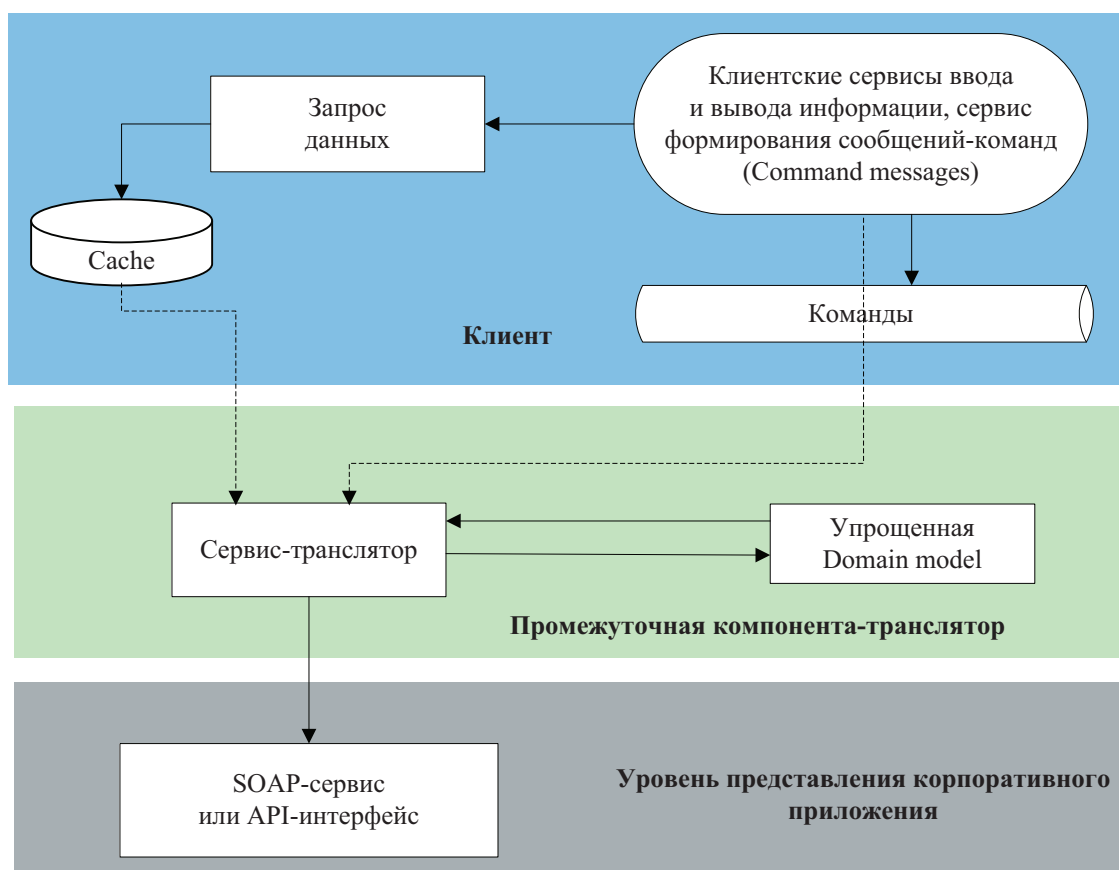


Рис. 17. Схема работы промежуточной компоненты с корпоративными приложениями, имеющими собственный API

К корпоративным приложениям третьего типа относятся приложения с полностью закрытым программным кодом, не имеющие интерфейса для коммуникации с внешними приложениями. Подобные приложения, как и приложения второго типа, составляют значительную долю современного рынка корпоративных информационных систем. Однако в последнее время сформировалась тенденция большей открытости программного кода и коммуникаций с внешними компонентами. Следует отметить, что включение мобильной клиентской части в подобную систему является наиболее трудоемким среди перечисленных.

Простая промежуточная компонента-транслятор, разработанная для использования в корпоративных системах с центром интеграции типа BizTalk, трансформируется в небольшой интеграционный сервер, имеющий связь с базой данных и файл-репозиторием, собственную авторизацию пользователей и собственную, не охватывающую весь функционал корпоративного приложения, логически не противоречащую ему область описания данных (рис. 18).

На современном рынке представлены немногочисленные информационные системы, не имеющие собственного API, у которых закрыт доступ к контейнеру базы данных (защищен паролем) или общение с файл-сервером происходит не непосредственно, а по специализированному каналу. Для подобных систем создание мобильной клиентской части невозможно.

**8. Перспективы развития и особенности взаимодействия мобильного клиентского приложения и корпоративного приложения.** Предлагаемая технология (см. пп. 2–6) предусматривает достаточно высокий уровень изолированности серверной и



Рис. 18. Схема работы промежуточной компоненты с полностью закрытыми корпоративными приложениями

мобильной клиентской частей приложения, что позволяет развивать их независимо друг от друга и минимизировать затраты на модернизацию всей системы в будущем. В частности, преимуществом методики является то, что при трехуровневой архитектуре серверная часть корпоративного приложения может быть подвергнута любым изменениям, обусловленным производственной необходимостью, но это не повлияет на работу мобильной клиентской части. Например, серверная часть может быть интегрирована с дополнительными сервисами и (или) служебными программами, разработанными третьими фирмами. Следует отметить, что при проектировании необходимо исключить варианты развития, которые могут оказать деструктивное влияние на области определения системы.

В случае использования в качестве корпоративного приложения решения с трехуровневой архитектурой API серверной части приложения может быть расширен, что не потребует доработки мобильного приложения.

Вследствие ограниченности вычислительных ресурсов устройства прежде чем выбрать направление развития, необходимо определить основное назначение данного мобильного приложения. В случае если основным назначением мобильного клиентского приложения яв-

ляется предоставление корпоративной информации для решения оперативных задач, наиболее вероятным направлением развития мобильного клиентского приложения будет совершенствование графического интерфейса, а также создание сервиса для более удобного поиска и представления информации. Если же основным назначением является анализ бизнес-ситуаций, возникающих при реализации того или иного проекта, то развитие будет происходить по пути адаптации модели, увеличения количества критериев анализа и совершенствования различных отчетов и диаграмм.

Еще одним перспективным направлением развития мобильного клиента является создание решения, в котором в качестве источников данных будут одновременно использованы различные сервисы корпоративного сервера. Например, имеется публичный online-сервис фотографий соискателей на определенную работу (например, социальная сеть Linked In). Существует корпоративное приложение, организующее поиск и сопровождение соискателей (например, рекрутинговая система TRIS [1]). При этом работа клиентской части программы, созданной для стационарного компьютера, не предполагает использования такого ресурса вследствие наличия закрытого программного кода, сложности имеющегося интерфейса и т. д. Для мобильного клиента можно расширить модель представления данных или добавить новый объект представления, в случае если отсутствует возможность изменить модель представления на сервере. В результате мобильный клиент приобретает новый уникальный (относительно настольного варианта) функционал, который будет востребован пользователями (в рассматриваемом примере это возможность просмотра фотографий).

Аналогично можно привести примеры развития функциональных возможностей для настольного клиента.

**Заключение.** Предложен и описан метод организации рабочего места пользователя корпоративной информационной системы, использующего в качестве рабочего компьютера современное мобильное устройство (iPad, iPod, netbook и т. д.). Особенностью этих устройств является ограниченность общих и вычислительных ресурсов, что, как правило, не позволяет использовать на них программы, разработанные для стационарных компьютеров. Предложенный метод решения данной задачи основан на использовании шаблона CQRS, что позволяет существенно снизить сетевой трафик; достаточно просто (с точки зрения реализации) применяются такие технологии, как Domain model [3] и Data Transfer Object [7]. Таким образом, решается проблема несоответствия между высокими требованиями, предъявляемыми к корпоративным системам, и ограниченными возможностями мобильных пользовательских устройств, на которые будет устанавливаться клиентская часть приложения.

Предлагаемый метод включает описание технологии организации мобильной работы с корпоративным приложением с учетом защиты передаваемых данных.

Использование предлагаемой методики позволяет работать с корпоративной системой как в случае стабильного сетевого соединения, так и в случае отсутствия сети. При наличии устойчивой связи с сервером в распоряжении мобильного пользователя находятся все ресурсы базы данных и все выполняемые им изменения немедленно сохраняются серверной частью приложения. Если связь с сервером прервана, пользователь работает автономно, используя хранилище данных на своем мобильном устройстве. Хранилище данных сохраняет информацию об объектах центральной БД в объеме, доступном на конкретном мобильном устройстве. Данные, полученные за время отсутствия связи, будут сохранены в центральной БД сразу после восстановления связи с сервером, причем метод позволяет проверять сохраняемые данные на совместимость и гарантирует сохранение их целостности.

Вследствие невысокой трудоемкости реализации метод предоставляет возможность дальнейшего развития и использования на современном рынке приложений для мобильных устройств.

В настоящее время данный метод проходит апробацию при разработке приложения Mobile TRIS (рабочее название) в компании Recruitment Systems Pty Ltd.

Метод может быть применен для любой стандартной корпоративной системы, поскольку использует универсальную технологию, не зависящую от особенностей проектирования и разработки “материнской” системы.

## Список литературы

1. Recruitment Systems Ltd. Официальный сайт, руководство пользователя: [Электрон. документ]. [http://ww2.recruitmentsystems.com.au/support/index.php?\\_m=downloads&\\_a=viewdownload&downloaditemid=6&nav=0,1](http://ww2.recruitmentsystems.com.au/support/index.php?_m=downloads&_a=viewdownload&downloaditemid=6&nav=0,1). Проверено 03.08.11 г.
2. МЕЙЕР Б. Методы программирования: В 2 т. Пер. с фр. Ю. А. Первина / Под ред. А. П. Ершова. М.: Мир, 1982.
3. ФАУЛЕР М. Архитектура корпоративных программных приложений. М.: Вильямс, 2006.
4. ГОСТ Р 34.10-2001 Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. Введ. 2001 г.
5. ДАЛ У. Структурное программирование = Structured Programming. 1-е изд. / У. Дал, Э. Дейкстра, К. Хоор. М.: Мир, 1975. 247 с.
6. ESKERSON N., WAYNE W. Three tire client/server architecture: achieving scalability, performance, and efficiency in client server applications // Open Information Systems. 1995. N 1. P. 3–20.
7. ГАММА Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влссидес. СПб.: Питер, 2007. 366 с.

*Платонов Юрий Георгиевич — асп. Института систем информатики СО РАН;  
тел.: 913-950-60-71; e-mail: y.platonov@mail.ru*

Дата поступления — 13.04.11