ООО «Интернет-агентство ИНТЕРВОЛГА»
ИНН: 3445103528

INTERVOLGA
компетентный веб-интегратор

+7 (495) 648-57-90
+7 (8442) 95-99-99

Аудит технического состояния корпоративного портала ___

июль 2018

Предмет аудита

Цель аудита – поиск мест в коде, вызывающих медленную работу портала, а также проблем безопасности.

Задача поиска логических ошибок и “точек улучшения” не ставилась.

Важный момент исследования — проводился анализ **копии** действующего корпоративного портала. Копия содержит идентичную БД и исходный код, но отличное от оригинала серверное окружение. Это накладывает некоторые условия:

1. Ограничения при работе с сервисами, отвечающими за интеграцию с телефонией (один из главных функциональных элементов данного портала).
2. Ненастроенное кэширование.

Модифицированные файлы ядра

Изменений, которые могли бы повлиять на производительность, в ядро не вносилось. Есть несколько измененных файлов, но они не могут быть причиной повышенной нагрузки на портал.

Пример измененного файла — **/bitrix/modules/main/lib/localization/loc.php**

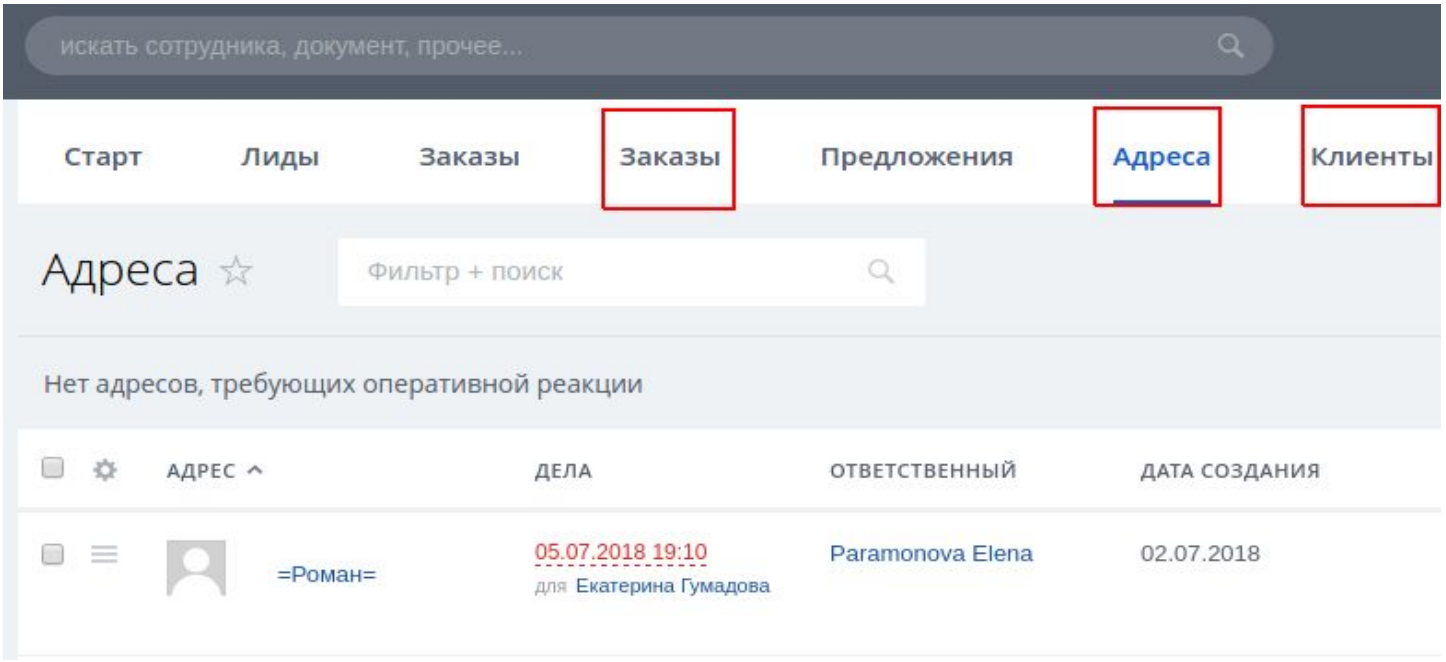
К результату стандартной функции **Loc::getMessage()**, которая используется при работе с языковыми файлами, применяется функция **lang_correct()** (определена в файле **/bitrix/php_interface/init.php**). Функция используется для замены слов во всех сообщениях на портале (Изображение 1):

- “Компания” на “Клиент”.
- “Контакт” на “Адрес”.
- “Счет” на “Заказ”

Это несколько странное решение, серьезных проблем оно не создает, но каждое последующее обновление портала будет перезаписывать внесенные в код изменения.
Решить вопрос можно следующим образом:

- Убрать функцию **lang_correct()** из ядра.
- Расширить функционал стандартной функции **Loc::getMessage()** с помощью кода, который будет расположен вне ядра.
- Указать системе использовать расширенный метод вместо стандартного.

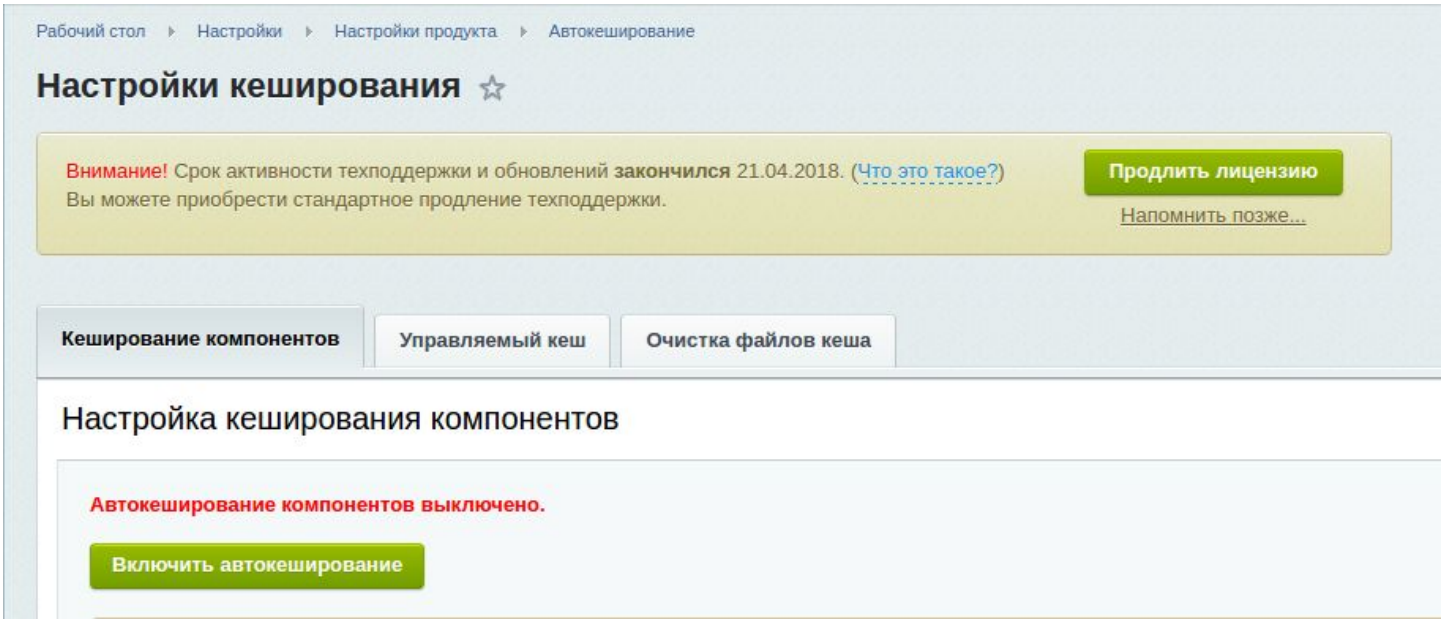
ИНТЕРВОЛГА обладает технологией intervolga.injections, которая позволяет решать такие задачи без правки кода ядра.



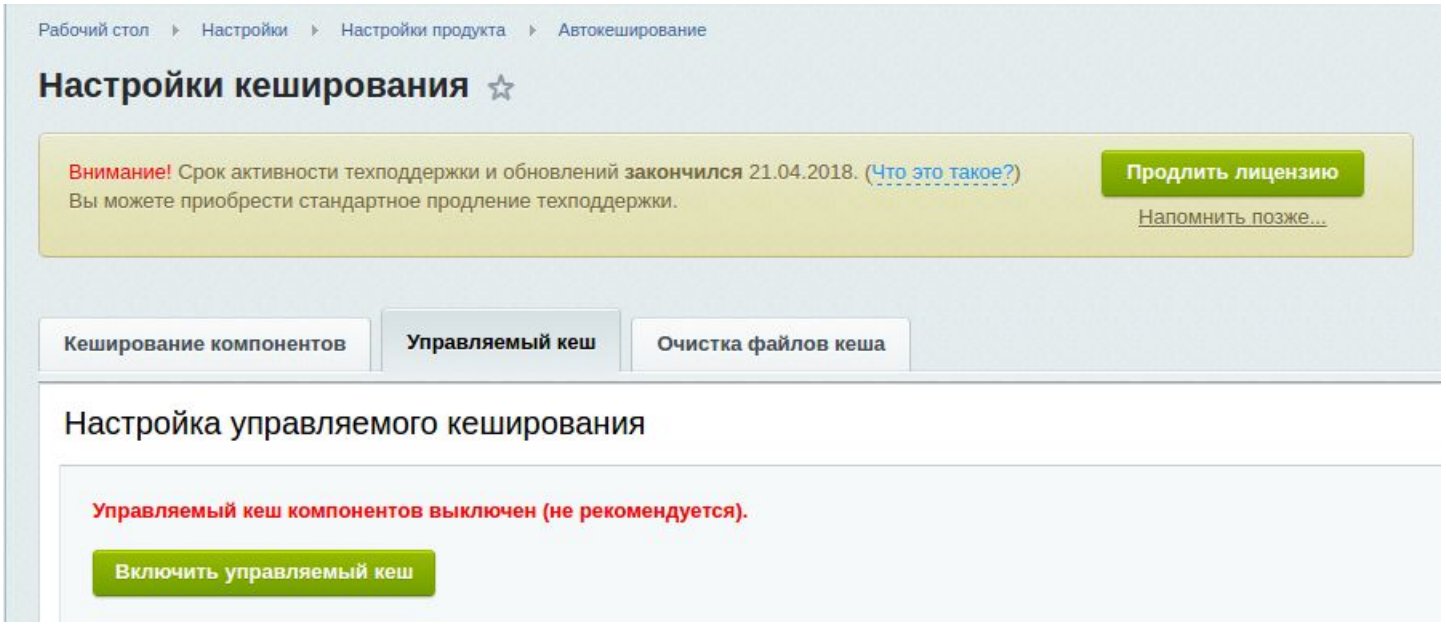
Изображение 1 — Результат работы функции **lang_correct()**

Кэширование

На предоставленном нам портале было отключено кэширование компонентов (Изображение 2) и тегированный кэш (Изображение 3).



Изображение 2 — Кэширование компонентов отключено



Изображение 3 — Тегированный кэш отключен

После активации настроек кэш всё равно не пришел в рабочее состояние.
Причина в следующем — в файлах bitrix/php_interface/dbconn.php и bitrix/.settings_extra.php есть явное указание на использование **memcache** вместо стандартного файлового кэша.

На тестовой версии портала, которая была нам предоставлена, **memcache** не настроен.
В том случае, если на действующем портале кэш также отключен или не настроен, то это может влиять на скорость работы сайта.

Анализ запросов к базе данных

Согласно монитору производительности самые долгие запросы — это запросы к странице /api/v2/index.php. Время отдельных запросов достигает 60 и более секунд (Изображение 4, 5).

Эти запросы создают 99% нагрузки.

20 самых нагружающих страниц	Ошибки разработки[2]	Нагрузка	Количество хитов	Среднее время (сек.)
/api/v2/index.php		99.04%	765	20.2621
/crm/contact/index.php		0.32%	120	0.4129
/include/ajax/data_orders_contact.php		0.16%	27	0.9272
/crm/invoice/index.php		0.13%	51	0.3870
/ruta/func/load_order_crm.php		0.07%	24	0.4638
/crm/lead/index.php		0.04%	23	0.2693
/crm/company/index.php		0.04%	32	0.1867
/bitrix/admin/1c_exchange.php		0.02%	22	0.1454
/company/personal.php		0.02%	15	0.1748

Изображение 4 — Отчет “Монитора производительности”

Хит	Страница-время
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=null&status=	110.3110
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=null&status=	106.3070
>> /crm/lead/list/?sessid=835c7b217c32d779c3f7e966dc9ea741&internal=true&grid_id=CR...	96.7248
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=null&status=	94.3111
>> /crm/lead/list/?sessid=835c7b217c32d779c3f7e966dc9ea741&internal=true&grid_id=CR...	88.2706
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=null&status=	86.6701
>> /crm/lead/list/?sessid=835c7b217c32d779c3f7e966dc9ea741&internal=true&grid_id=CR...	86.5094
>> /crm/lead/list/?sessid=835c7b217c32d779c3f7e966dc9ea741&internal=true&grid_id=CR...	84.4948
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=null&status=	73.2759
>> /bitrix/components/bitrix/im.messenger/im.ajax.php?UPDATE_STATE&V=104	63.1919
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=256&status=startCall	61.8050
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=268&status=startWaiting	61.1349
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=256&status=startCall	60.5645

Изображение 5 — Самые долго исполняемые запросы, данные действующего портала

API, расположенное в данном разделе, отвечает за интеграцию портала с телефонией.

Согласно отчету о производительности проблемным является только метод **checkOktellOperatorStatus**.

Метод получает данные из таблицы, описывающей текущие статусы операторов ("в ожидании", "занят", ...).
При попытке получить данные об операторе, ID которого нет в таблице, метод будет повторять попытки получить данные в течение 60 секунд (с перерывами на 0.001 секунды между запросами). За это время будет совершено около 40 000 запросов к базе, что недопустимо много.

Такая реализация, видимо, была выбрана потому, что элементы в таблице постоянно обновляются. Поэтому нужный элемент может быть получен всего через несколько секунд после начала работы метода. Даже если ранее его там не было.

Около половины запросов, которые заняли 60 секунд — запросы с пустым параметром **phoneInner** (Изображение 6). Такие запросы будут выполняться в течение всех 60 секунд, т.к. в таблице не может находиться элемент с пустым значением поля **phoneInner**.

Хит	Страница-время
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=797&status=startWaiting	60.0309
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=372&status=startWaiting	60.0305
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=null&status=	60.0304
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=null&status=	60.0303
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=null&status=	60.0299
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=211&status=	60.0299
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=372&status=startWaiting	60.0298
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=null&status=	60.0297
>> /api/v2/?r=phone/checkOktellOperatorStatus&phoneInner=797&status=startWaiting	60.0296

Изображение 6 — Пустые параметры в запросах к методу **checkOktellOperatorStatus**

Источник запросов к методу **checkOktellOperatorStatus** — не портал, в его коде нет таких вызовов.

Для снижения нагрузки на систему и исправления ситуации необходимо:

1. Узнать, какая система посылает на портал такие запросы и исключить запросы с пустым параметром **phoneInner.**
2. Возможно, стоит модифицировать сам механизм работы метода. В любом случае 40 000 запросов к базе — гораздо больше, чем нужно.

Возможные варианты модификации работы метода (отсортированы по возрастанию затрат на реализацию)

1. Увеличить время ожидания между повторными запросами к базе, установив значение равным от 1 до нескольких секунд.
    a. **Преимущества** — не требуется дополнительного анализа механизма работы API
    b. **Недостатки** — в худшем случае страница всё равно будет обрабатываться 60 секунд, занимая ресурсы сервера.
2. Вместо цикла длиной 60 секунд делать один запрос, к БД и либо возвращать найденные данные, либо сообщение об отсутствии данных.
    a. **Преимущества** — сервер не будет в течении минуты работать с потенциально ненужным процессом.
    b. **Недостатки** — требуется внесение изменений в систему, которая посылает запросы на портал. Изменения потенциально небольшие, но требуют доступа к этой системе и некоторого анализа кода на ее стороне.
3. Пересмотреть вопрос о целесообразности хранения некоторых данных непосредственно на портале.
    a. **Преимущества** — часть сложной логики может удалена с портала, что облегчит сопровождение сайта в дальнейшем.
    b. **Недостатки** — требуется анализ архитектуры взаимодействия портала с телефонией. Скорее всего текущий вариант работы был принят не безосновательно. Могут присутствовать не очевидные на данном этапе моменты, которые мы не можем предвидеть, работая с тестовой копией портала.

Безопасность

На нескольких страницах сайта есть код, который позволяет авторизоваться из-под пользователя с правами администратора.

Например:

1. Переходим на страницу ___/local/ch1.php (Изображение 7)
2. После этого мы сразу авторизованы с правами администратора (Изображение 8).
3. Код с уязвимостью — Изображение 9.

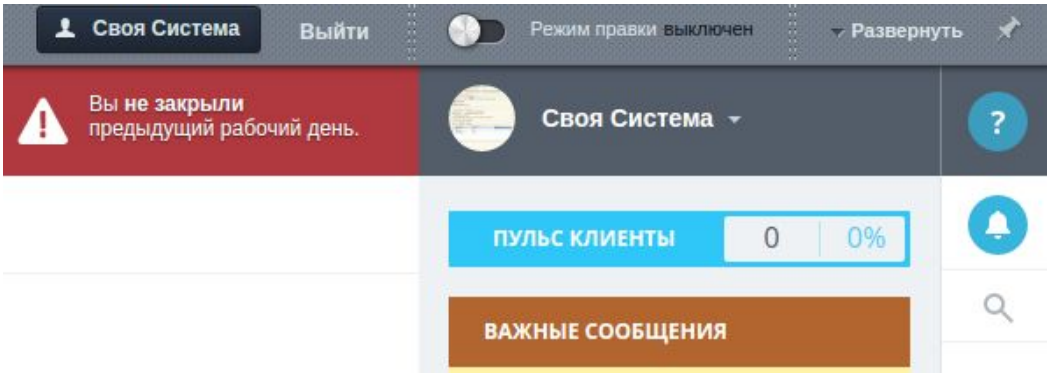
___

Изображение 7 — Страница, содержащая уязвимость

Такие страницы можно разделить на несколько типов

1. Не участвуют в работе портала. Забыли удалить в процессе разработки.
2. Страницы, к которым обращается cron (на нашем сервере cron не работает).
    a. Они должны быть как минимум закрыты от доступа по http (на тесте не реализовано). Если такая защита реализована на бою — проблем быть не должно.
    b. Некоторые методы из API Bitrix’а требуют, чтобы пользователь, их вызывающий, был авторизован. Не факт, что авторизация нужна на всех тех страницах, на которых она присутствует, но, как минимум, можно было бы разлогинить пользователя в конце скрипта.

3. Есть страницы, видимо предназначенные для использования внешними сервисами — обрабатывают входные параметры, отдают информацию в формате JSON. Замечания к таким страницам аналогичны предыдущему пункту.



Изображение 8 — После посещения страницы с уязвимостью мы авторизованы на портале

```
@set_time_limit(0);
@ignore_user_abort(true);


global $USER;
$USER->Authorize(505);

if (!CModule::IncludeModule('crm'))
{
  ShowError(GetMessage('CRM_MODULE_NOT_INSTALLED'));
  return;
}
```

Изображение 9

Список проблемных страниц

1. ./tmp/filter_adresses/ — несколько страниц в разделе
2. ./tmp/test_ch_timer/ — несколько страниц в разделе
3. ./tmp/test/ — несколько страниц в разделе
4. ./api/v2/bootstrap.php.v0
5. ./api/v2/bootstrap.php
6. ./local/ch_timer.php
7. ./local/ch1.php
8. ./auth/alt.php
9. ./upload/timer_to_crm.php
10. ./ruta/func/ — несколько страниц в разделе
11. ./ruta.2018-06-08/func/ — несколько страниц в разделе
12. ./ruta.2018-06-08-updated/func/ — несколько страниц в разделе

База данных и оперативная память сервера

Проблема:

```
До работы операторов висели процессы скл и забирали памяти 5 гбт. С учётом того что никто ещё не работал.
После перезагрузки скл службы процессы скинулись.
mysql> SHOW FULL PROCESSLIST;
+-------+--------+----------------------+----------+---------+------+------------------+------------------
-----------------------------------------------+-----------+---------------+
| Id    | User   | Host                 | db       | Command | Time | State            | Info
| Rows_sent | Rows_examined |
+-------+--------+----------------------+----------+---------+------+------------------+------------------
-----------------------------------------------+-----------+---------------+
| _____                                                    |         0 |             0 |
```

Наличие некоторого количества спящих процессов в БД при работе Bitrix'а вполне допустимо.
В случаях, если количество таких запросов становится проблемой, то рекомендуется

- Установить значение константы **DBPersistent** равным FALSE (файл **/bitrix/php_interface/dbconn.php**).
- В файле **/bitrix/.settings.php** в секции **connections** установить параметр **options** равным 2.

На портале установлены именно эти параметры.

В ходе тестирования выявлено, что после обращения к методу API по адресу
/api/v2/?r=phone/checkOktellOperatorStatus&phoneInner…
в sql появляется один новый процесс, который преимущественно находится в состоянии SLEEP (Изображение 10), периодически переходя в состояние QUERY (Изображение 11).

Изображение 10

Изображение 11

После окончания работы метода процесс завершается.

Согласно результатам тестирования, за 60 секунд метод **checkOktellOperatorStatus** посылает 40 000 запросов к БД (именно такая большая частота запросов объясняет то, что списке процессов mysql у вышеуказанного процесса в колонке "Time" значение постоянно равно 0).
Обращения к разделу /api/v2/ происходят достаточно часто (замеры с боя — 760 хитов за 10 минут, более 1 хита в секунду). Около половины запросов к этому разделу приходятся на метод **checkOktellOperatorStatus**.
Именно такая большая частота запросов создает большое количество запросов в БД, которые в свою очередь нагружают оперативную память.
Работу данного метода необходимо исправить, конкретные предложения описаны выше в специальном разделе.

Измененные компоненты и шаблоны

Теоретическая вводная.
Обновление версии Bitrix — стандартная, полуавтоматическая, регулярно выполняемая процедура.

Представляет собой модификацию файлов, находящихся в разделе **/bitrix/** (ядро проекта, шаблоны, компоненты).
Выполняется по инициативе администратора сайта, но полностью производится программным обеспечением Bitrix'a.

Разработчикам сайта категорически не рекомендуется вносить изменения в любые файлы, содержащиеся в директории **/bitrix/** (за исключением нескольких разрешенных).
Минимальный риск при невыполнении этого условия — внесенные правки могут быть перезаписаны при последующем обновлении.
При самом плохом варианте развития событий после обновления сайта часть функционала перестанет работать. Логика работы отдельных функциональных компонентов может непредсказуемо измениться.

Минимальной структурной частью проекта под управлением Bitrix является **компонент**.
Система предоставляет большую коллекцию предустановленных компонентов, которые могут использоваться как отдельно, так и во взаимодействии с другими компонентами.

Каждый компонент может содержать несколько **шаблонов компонента**. Шаблон определяет отображение данных сформированных компонентом.

При доработке портала часто возникает задача изменить стандартную логику работы компонентов. Решить такую задачу можно несколькими вариантами, один из которых — указать системе использовать новые компоненты вместо предустановленных.

Самый очевидный способ добиться этого — скопировать компонент в раздел **/local/**. Упрощенная схема, по которой Bitrix выбирает компонент для использования:

- Производится поиск компонента в разделе **/local/**. Если поиск выполнен успешно, то подключается именно он.
- Иначе поиск компонента происходит в разделе **/bitrix/**

Такой способ модификации стандартных компонентов встречается с проблемами при обновлении сайта.

1. Портал не получает доступ к части обновленных данных, т.к. продолжает использовать код из раздела /local/, который остался неизменным.
2. Многие компоненты связаны с другими. Такие связи могут быть изменены в результате обновления. Если в разделе **/local/** находится только часть из взаимодействующих между собой компонентов, то с вероятностью близкой к 100% это приведет к ошибкам, которые достаточно трудно диагностировать.

Сейчас на портале в папке /local/ кроме прочего находится

1. Более 20 переопределенных стандартных компонентов Bitrix'a.
2. Более 50 переопределенных шаблонов.

При таком их количестве обновление портала крайне затруднено.

Некоторые шаблоны и компоненты в папке **/local/** ничем не отличаются от аналогичных в папке **/bitrix/**, их по возможности нужно удалить.

Примеры:

1. “crm.activity.calendar.add” (/local/components/bitrix/crm.activity.calendar.add/)
2. “crm.activity.calendar.list” (/local/components/bitrix/crm.activity.calendar.list/)

Примеры измененных компонентов:

1. “crm.invoice.edit”, страница редактирования заказа (/crm/invoice/edit/номер_заказа/). Изменения:
    a. Логика формирования цены
    b. Вывод системных ошибок
2. “crm.contact.edit”, страница редактирования контакта (/crm/contact/edit/номер_контакта/). Изменения:
    a. Логика проверки региона для реквизитов.
    b. Проверка дублирующихся элементов.

Примеры измененных шаблонов:

1. компонент “crm.contact.edit”, шаблон “.default”, страница редактирования контакта (/crm/contact/edit/номер_контакта/). Несколько изменений, самое значительное из которых — добавление карты.
2. компонент “crm.contact.show”’, шаблон “.default”, детальная страница контакта (/crm/contact/show/номер_контакта/). Самые значительные изменения — дополнительные поля и вкладки.

Если перед разработчиками ставится задача обновления портала с большим количеством компонентов и шаблонов в папке /local/, то можно выделить два основных пути:

Поддерживать существующую структуру.
Исходный код модифицированных шаблонов и компонентов в папке /local/ остается нетронутым. Обновление выглядит следующим образом:

1. С помощью системы контроля версий (например Git) отслеживать изменения в папке /bitrix/.
2. Вручную анализировать, затрагивают ли они модифицированные элементы.
3. При необходимости подстраивать логику собственных шаблонов и компонентов под новую версию ядра.

Главный недостаток — требуется тратить время на анализ кода и возможные правки в логике при каждом обновлении.
Изменения могут быть достаточно обширными, затрагивать большое количество файлов. Задача синхронизации нового ядра с существующими доработками может занять от нескольких часов до нескольких дней.

Придерживаться подхода, не препятствующего обновлению.
Существует несколько способов доработки интерфейса портала, которые не создают проблем при обновлении. Некоторые из них:

1. Использование отложенных функций.
2. Собственные типы пользовательских полей.
3. Модификации на стороне клиента (JavaScript).
4. Стандартные события Bitrix’a.
5. Регистрация собственных событий. Позволяет манипулировать данными без непосредственного

доступа к исходному коду компонентов.

Преимущества подхода — нет необходимости отслеживать список изменений каждого обновления.

Портал содержит большое количество доработок. Для того, чтобы реализовать их на вышеперечисленных принципах, необходим более детальный анализ модифицированных частей сайта. Такая задача может занять несколько десятков часов.

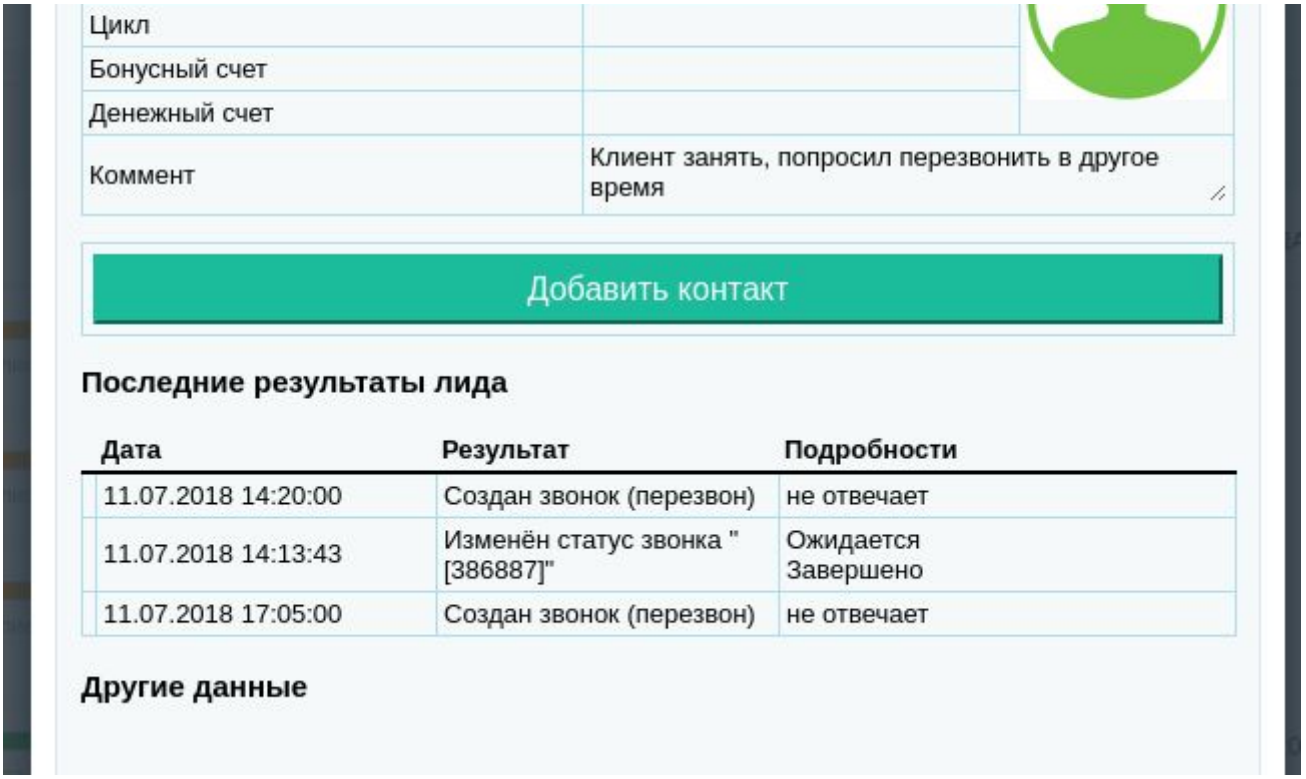
Самые нагруженные страницы

Страница **/api/v2/index.php** описана выше.

Согласно отчёту монитора производительности (Изображение 4) и отдельным замерам самыми нагруженными страницами являются следующие

1. **/crm/contact/index.php** — страница со списком контактов (“адреса” в терминах текущего портала), среднее время ответа 0.41 с.
   Путь до страницы: Меню -> CRM -> Адреса.
   На странице расположен предустановленный компонент **crm.contact.list** с модифицированным шаблоном. От стандартного отличается тем, что в него добавлена опция “быстрый просмотр”.
   Запросов, излишне нагружающих портал, не имеет.

2. **/include/ajax/data_orders_contact.php** — таблица с последними результатами при быстром просмотре лида или контакта (Изображение 12), среднее время 0.93 с.
   На странице производится выборка данных из различных таблиц (компании, контакты, сделки).
   Запросы к БД, выполняемые данной страницей, можно оптимизировать.



Изображение 12 — Быстрый просмотр лида

3. **/crm/invoice/index.php** — страница со списком заказов, среднее время ответа 0.38 с.
   Путь до страницы: Меню -> CRM -> Заказы.
   Стандартный компонент **crm.invoice.list**
4. **/ruta/func/load_order_crm.php** — среднее время 0.46 с.
   Выбирает из БД и отдает данные о заказах в формате JSON. Предположительно должен использоваться внешними сервисами.
5. **/crm/lead/index.php** — страница со списком лидов, среднее время 0.27 с.
   Путь до страницы: Меню -> CRM -> Лиды.
   Компонент **crm.lead.list** с пользовательским шаблоном. Шаблон не создает дополнительной нагрузки.

Ни одна из вышеперечисленных страниц с точки зрения производительности не является критичной.

Рекомендации по оптимизации

В различных частях портала присутствует программный код, выполняющий лишние запросы к БД. При небольшом объеме данных проблема может быть не очевидна. Но по мере увеличения контента на портале страницы, совершающие такие запросы, могут загружаться больше обычного и создавать нагрузку на БД, которую можно было бы избежать.
Примеры таких файлов:

1. **/api/v2/bin/commands/AddressCommand.php**, функция **geocode()**, используется в процессе геокодирования адресов из Яндекса.
2. **/api/v2/bin/commands/TelephonyCommand.php**, функция **processPhones()**, используется при формировании списка звонков, перезаписывает таблицу **bdialer_oktell**.
3. **/local/php_interface/init.php**, функция **GetCrmHistory()**.
4. **/local/php_interface/init.php**, функция **getDeliviriedOrdersProduct()**.
5. **/include/ajax/data_orders_contact.php**, формирует таблицу с последними результатами при быстром просмотре лида или контакта.
6. **/include/ajax/show_lead.php**, используется при формировании информации и лиде или контакте.

Итог

Корпоративный портал имеет значительные доработки, что делает невозможным его обновление посредством стандартного функционала. При обновлении при текущей ситуации возникнут ошибки.

Задача	**Оценка, ч**	**Комментарий**
Исследовать возможность приведения портала к принципам разработки, позволяющим обновлять корпоративный портал посредством стандартного функционала 1С-Битрикс.	10	Включает работы: 1. Детальный анализ модифицированных частей сайта. 2. Оценка оптимизации.

Установка GIT на сервер	От 5 часов, требуется изучение действующей архитектуры.	Потребуется установить систему контроля версий (например ГИТ) на сервер.
Услуги SLA. Мониторинг доступности корпоративного портала, проверка кода сторонних разработчиков на вредоносные изменения, на соответствие стандартам разработки и на отсутствие дублирующих запросов.	15 000 — 40 000 руб в месяц	Проверка кода 1 раз в две недели, 6-12 часов технического специалиста в месяц (в зависимости от объема кода) Проверка сайта сканером безопасности, монитором производительности и сканером троянов 1С-Битрикс 1 раз в 2 недели. Под вопросом функциональное тестирование и бекапирование портала. Нужно обсудить ваши задачи и требуемые сценарии. Готовы взять на себя ВСЕ работы по порталу.
Включить или настроить memcache	8	8+ с учётом технических трудностей
Переделать механизм работы с методом **checkOktellOperatorStatus**		Требуется доступ в систему и контакт с администраторами телефонии для детальной оценки.
Удалить или закрыть найденные дыры в безопасности	6	Около 30 файлов, оценка для всех.
Исправить программный код, выполняющий лишние запросы к БД	10	Оценка дана на исправление шести файлов: - **/api/v2/bin/commands/AddressCommand.php**, функция **geocode()** - **/api/v2/bin/commands/TelephonyCommand.php**, функция **processPhones()**, **bdialer_oktell**. - **/local/php_interface/init.php**, функция **GetCrmHistory()**. - **/local/php_interface/init.php**, функция **getDeliviriedOrdersProduct()**. - **/include/ajax/data_orders_contact.php** - **/include/ajax/show_lead.php**,

___, ведущий программист **___**

INTERVOLGA
компетентный веб-интегратор

Степан Овчинников,
ген.директор, к.т.н.

8-903-316-03-10
stepan@intervolga.ru

MVP 1С-Битрикс