**BUS factor или “один - плохое число для бизнеса”**

Для тех, кому не знакомо понятие **bus factor**, обратимся к первому тезису из wiki, в котором говорится, что Бас фактор - это фактор, показывающий количество участников проекта, после потери которых (в оригинале — «попадания» которых под автобус), проект не сможет быть завершен оставшимися участниками.
Вы также могли встречать понятие **truck factor (фактор грузовика) или brick factor (фактор кирпича)**, но суть от этого не меняется.
Если перевести с айтишного на человеко-понятный язык, **bus factor - это про «а что случится с вашим бизнесом/проектом, если вы лишитесь вот этого конкретного участника или команды,  в руках или умах которых содержится  90-100% информации для работоспособности проекта?**

Отвечая на этот вопрос, мы можем увидеть зоны уязвимости проекта и подстелить соломку там, где ценная информация, приводящая в действие движение нашего проекта, сосредоточена в руках одного лица.

**Самое плохое число для бизнеса - это один:**
- один канал продаж (может отвалиться или стать неэффективным одномоментно и придется потратить уйму времени, чтоб настроить дополнительные источники получения трафика;;
- один клиент, пусть и крупный (если он уйдет, команда будет у разбитого корыта);
- один человек, который знает о том, как устроены все бизнес-процессы;
- один разработчик, у которого лежат все доступы, и они принадлежат не вам, и ваш бизнес в буквальном смысле зависите от того, чтоб он был онлайн был контактным;
- один разработчик, который разрабатывал вам ПО, но не оставил никакой технической документации, и никто, кроме него не знает как все реализовано и будет тратить дни, недели, или месяцы, чтобы разобраться в логике происходящего.
Продолжать можно бесконечно.

Суть в другом: если ваш бизнес или проект имеет одну или более зон уязвимости, где ресурс сосредоточен в руках одного специалиста, у вас высокий уровень бас фактора.
Бас фактором может выступить любое событие, от “попал под автобус”, до “внезапно заболел” и не вышел на связь тогда, когда был критично нужен.
Простой способ оцифровать ваш bus factor - это взять количество людей, задействованных на проекте. Раз уж мы пишем эту статью в блоге студии веб-разработки, давайте возьмем сценарий из этой области.
Допустим, у вас 7 человек задействовано на проекте, из них 1 проджект-менеджер, 1 бизнес-аналитик, 3 разработчика занимаются бекендом и пишут код, также есть 1 фронтент и 1 дизайнер. Бас фактор в контексте разработки ПО в этом проекте равен трем, потому как ключевая информация рассредоточена между 3-мя разработчиками бекенда.
Если у вас небольшая компания, и вы работаете с фрилансерами или индивидуальными разработчиками - ваш уровень бас-фактора также равен одному.
И здесь уже должно становиться более волнительно :)

**5 базовых шагов как вы можете снизить риски  высокого bus factor проекта, если вы заказчик:**

1. на регистрации домена на вас, а не на разработчика/маркетолога;
2. настоять на регистрации хостинга на вас, а не на разработчика, и уже из-под вашего аккаунта раздавать доступы всем, кому это будет необходимо;
3. настоять на регистрации сервера на вас, а не на разработчика, и аналогично выдавать доступы другим участникам проекта, а не зависеть от кого-то извне
4. договориться о работе по ТЗ (техническому заданию), чтобы любой другой специалист мог разобраться в том, что вы делали “до” того, как привлекли к проекту кого-то еще;
5. попросить составлять техническую документацию, особенно касается сложных архитектурных проектов, которые будут иметь вложенную структуру связей  и большое количество доработок функционала в будущем.

Стоит отметить, что работа с составлением ТЗ и технической документации является более трудоемкой для специалиста или команды, в контексте нагрузки по человеко-часам работы, а значит и будет немного дороже стоить для вас как заказчика.
Однако, это тот самый случай, когда проявив должный уровень уважения и внимания сегодня - вы сэкономите больше времени и денег компании завтра, послезавтра и годы спустя.

**7+1 базовых шагов как вы можете снизить риски  высокого bus factor проекта, если вы и есть та самая команда разработчиков или индивидуальный специалист:**

       Пункт № 0. На каждый хоть сколько-нибудь нужный проект должен быть репозиторий с описанием как разворачивать проект.

1. Создавать базу знаний и техническую документацию по проекту, и хранить ее в облаке компании, а не в персональных облаках отдельных разработчиков, и уж тем более не локальных файлах на ноутбуке или компьютере отдельных специалистов;
2. Репозиторий должен быть доступен более чем у одного разработчика со всеми полномочиями;
3. Делайте регулярные бекапы важных данных, желательно не на том же сервере, где и продакшн.
4. Систематизируйте требования к документированию кода, неймингу классов и переменных, к расположению модулей в проекте и прочему. Чем больше моментов вы стандартизируете - тем меньше будет уровень бас фактора на проекте с любым числом разработчиков.
5. Делайте код-ревью всех разработчиков, даже тех, кто пишет “как Боженька” (особенно, если он пишет “в своем стиле”.
6. Если в проекте есть закомментированные куски кода, всегда указывайте почему они закомментированы и в каких случаях их нельзя трогать и раскомментировать.
7. Если какую-то часть работ (допустим CI/CD) настраивал один специалист, и никто не вдавался в детали как он его настроил - попросите составить его техническую документацию, чтоб не обращаться к нему каждый раз когда необходимо изменить pipeline.

Подитожив вышесказанное, хочется добавить, что bus factor есть абсолютно во всех проектах и бизнесах. Вопрос только в том, насколько он высокий или низкий.
Как правило, небольшим компаниям и командам присущ более критичный уровень бас фактора, в силу сосредоточения большого количества информации именно в руках отдельных людей. По мере роста бизнеса и количества людей, задействованных в реализации поставленных задач, бас фактор становится ниже.
Однако, время от времени любой проект требует “ревизии” на предмет уязвимости в контексте bus factor.

**The “BUS” factor or “one is a bad number for business”**

For those who are not familiar with the concept of the bus factor. Let's refer to the first thesis from the “wiki”, which says that the “bus” factor is a factor showing the number of project participants, after the loss of which (in the original – which was "run over” by a bus), the project cannot be completed by the remaining participants.

You may also have come across the concept of the “truck” factor or “brick” factor, but this does not change the essence.

Being translated from the IT language into the human-understandable one, the "bus" factor is about “what will happen to your business/project if you lose this particular member or team, whose hands or minds contain 90-100% of the information for the project to work?

By answering this question, we can see the vulnerability areas of the project and cushion a shock where the valuable information that drives our project is concentrated in the hands of one person.

**The worst number for business is “one”:**

- one sales channel (may fall off or become ineffective at once and you will have to spend a lot of time to set up additional sources of traffic;

- one client, albeit a large one (if he leaves, the team will end up with nothing);

- one person who knows how all the business processes work;

- one developer who has all the accesses, and they do not belong to you, and your business literally depends on the fact that he is online and has a contact;

- one developer who developed software for you, but did not leave any technical documentation, and no one except him knows how everything is implemented, and you will spend days, weeks, or months to figure out the logic of what is happening.

You can continue indefinitely.

The point is different: if your business or project has one or more areas of vulnerability, where the resource is concentrated in the hands of one specialist, you have a high level of the “bus” factor.

Any event can turn out to be the "bus" factor, from “was run over by a bus” to “suddenly fell ill” and did not get in touch when you need him critically.

An easy way to digitize your “bus” factor is to take the number of people involved in the project. Since we are writing this article on the blog of the web development studio, let's take a scenario from this area.

Let's say you have 7 people involved in the project, 1 of them is a project manager, 1 is a business analyst, 3 developers are engaged in the "backend" and write a code, there are also 1 "frontend" developer and 1 designer. The "bus" factor in the context of software development in this project equal to 3, because the key information is dispersed among three "backend" developers.

If you have a small company and you work with freelancers or individual developers your “bus” factor level is also equal to 1.

And this is where things get more exciting :)

**5 basic steps how you can reduce the risks of a high bus factor of a project if you are a customer:**

1. to insist on registering the domain for you, and not for the developer/marketer;
2. to insist on registering hosting for you, and not for the developer, and distribute access from under your account to everyone who needs it;
3. to insist on registering the server for you, and not for the developer, and similarly give access to other project participants, and not depend on someone outside;
4. to agree on work on the TOR (terms of reference) so that any other specialist can figure out what you did “before” you attracted someone else to the project;
5. to ask for technical documentation, especially for complex architectural projects that will have a nested structure of relationships and a large number of functional improvements in the future.

It is worth noting that the work with the preparation of technical specifications and technical documentation is more time-consuming for a specialist or a team in the context of the workload in terms of person-hours of work, which means it will cost a little more for you as a customer.

However, this is the case when by showing the proper level of respect and attention today you will save more time and money for the company tomorrow, the day after tomorrow and years later.

**7 + 1 basic steps how you can reduce the risks of the high "bus" factor of the project, if exactly you are that development team or an individual specialist:**

Point № 0. For every project that is at least somewhat necessary there should be a repository with a description of how to deploy the project.

1. Create a knowledge base and technical documentation for the project, and store it in the company's cloud, and not in the personal clouds of individual developers, and even more so not in local files on a laptop or computer of individual specialists;
2. The repository must be available from more than one developer with full authority;
3. Make regular backups of important data, preferably not on the same server as the production one.
4. Systematize the requirements for documenting code, naming classes and variables, the location of modules in the project, and so on. The more points you standardize, the lower the "bus" factor will be on a project with any number of developers.
5. Do a code review of all developers, even those who write “like God” (especially if they write “in their own style”.
6. If the project has any commented pieces of the code always specify, why they are commented out and in what cases they cannot be changed and uncommented.
7. If some part of the work (for example, CI / CD) was configured by one specialist, and no one went into details of how he configured it, ask him to draw up his technical documentation so as not to contact him every time you need to change the pipeline.

Summing up the above, I would like to add that the "bus" factor exists in all projects and businesses absolutely. The only point is how high or low it is.

As a rule, small companies and teams have a more critical level of "bus" factor due to the concentration of a large amount of information in the hands of individuals. As the business and the number of people involved in the implementation of the tasks, those were set, grow the "bus" factor becomes lower.

However, from time to time any project requires a “revision” for a vulnerability in the context of the "bus" factor.