В этом файле будут описаны направление развития и проблемы безопасности сервиса.

В качестве отправной точки выбраны OWASP Top-10 2023, далее акцент будет смещён на проблемы, фактически выявленные при использовании сервиса.

1. API1:2023 - Некорректная авторизация на уровне объекта

В большинстве API-вызовов пользователь только читает данные об объекте (общедоступные), без возможности редактирования. Возможность как-либо изменить данные касается:

* Добавления заявки
* Редактирования заявки
* Удаления заявки
* Редактирования проекта представителем команды

Для этих случаев нужно удостовериться, что проводятся соответствующие проверки уровня прав доступа.

1. API2:2023 - Некорректная аутентификация пользователей  
   Для аутентификации используются подписанные JWT-токены. Идентификация и первоначальная аутентификация производится провайдером SSO. С учётом этого, отпадает необходимость в множестве мер безопасности, связанных с паролями (проверка на надёжность пароля пользователя при регистрации, хранение паролей БД в безопасном зашифрованном виде, и т.д.).  
   JWT-токены хранятся в secure httpOnly cookies. Это даёт некоторую уверенность в невозможности их кражи через XSS-атаки. При этом всё же стоит пройтись по текстовым полям, доступным для ввода пользователям, и удостовериться в их экранировании, для предотвращения возможных XSS-атак.  
   Авторизация производится явной проверкой отношений объектов в БД, механизм индивидуален для каждого вида запроса.   
   Направления развития:

* Более осмысленное использование refresh-токенов. Сейчас при запросе на обновление access-токена проверяется лишь существование в БД refresh-токена для пользователя и действительность access-токена. То есть по факту наличие refresh-токена в таком виде неоправданно.   
  Следует хранить в cookies также и refresh-токен, проверять его соответствие с существующим токеном в БД и добавить модуль возможности завершить все активные сессии, удаляющий/переписывающий refresh-токен в БД.
* Структурное изменение механизма авторизации. Может быть полезно, если возникнет тенденция усложнения связей, которые будет неэффективно исследовать при обработке каждого запроса. Возможным недостатком станет увеличение зоны ответственности информации, хранящейся у пользователя (на данный момент в cookies хранятся только данные, позволяющие идентифицировать пользователя, без какой-либо информации о его правах). В этом случае нужно будет уделить особое внимание атакам, связанным с изменением соответствующих cookie.

1. API3:2023 - Нарушенная авторизация на уровне свойств объекта.

Кажется одной из более актуальных задач на данный момент.

Уже сейчас есть информация, которую не хочется передавать всем пользователям, например, список команд, подавших заявку на проект (хотим ограничиться только числом заявок). При этом подробная информация нужна для корректного пользования личным кабинетом (нужна подробная информация о каждой заявке команд пользователя).  
Уже реализован отдельный слой, преобразующий данные, полученные из strapi, в формат, используемый в бизнес-логике сервиса. На данный момент его задача - преобразование форматов, но на этом же уровне удобно ввести и механизмы авторизации.

1. API4:2023 - Неограниченное потребление ресурсов.

Существует возможная проблема, касающаяся потребления памяти и CPU серверов, связанная с DoS-атаками. Для её решения в дальнейшем будет внедрён ограничитель трафика, вид и параметры которого будут определены позже (отправной точкой может стать книга: “System Design”, Алекс Сюй, Глава 4).

На данный момент также очевидна проблема, касающаяся потребления хранилища данных при отправке заявок и результатов заполнения анкет:

Анкеты:

При поступлении новых результатов со стороны пользователя старые результаты не удаляются. Это сделано, чтобы злоумышленники не переписали данные заполнения анкет других людей (принадлежность результатов пользователю определяется по полю “почта” или “ФИО”). С учётом этого, можно вручную удалить данные, заполненные злоумышленниками, без потери действительных данных пользователя и без необходимости отката БД.  
При этом возникает проблема: злоумышленник может заспамить сервер результатами, перегрузив хранилище.

Не является приоритетной задачей по нескольким причинам:

1. Перед сохранением результатов проверяется наличие и правильность специального API-токена, определённого в настройках анкеты внутри Google Forms. Таким образом, программно заспамить результатами заполнения анкет у злоумышленника не получится, он будет вынужден этим заниматься вручную через UI Google Forms.
2. Использование Google Forms - временное решение. Проблема отпадёт сама собой, когда будет использован более проработанный модуль для работы с анкетами, использующий внутреннюю авторизацию сервиса или SSO СПбГУ. Это исключит/усложнит возможность злоумышленника выдавать себя за других пользователей.

Заявки:

Та же проблема, что и с анкетами: возможность спама. Есть два возможных сценария:

1. Пользователь выдаёт себя за представителя некоторой команды и пытается переполнить хранилище данных.   
   Этот вариант наименее вероятный, потому что при каждом запросе используются данные secure httpOnly cookie, идентифицирующие, аутентифицирующие и авторизующие пользователя. Возможности получить доступ к ним у злоумышленника слишком ограниченные, чтобы решение этой задачи безопасности было приоритетным.
2. Злоумышленник - действительный представитель команды. В этом случае система не запретит ему добавлять файлы.

Меры, которые необходимо предпринять:

* Ограничение числа файлов, передаваемых в одной заявке
* Ограничение размера каждого файла, передаваемого в заявке
* Ограничение форматов файлов, доступных для загрузки (технически не относится к пункту API4:2023, но касается того же модуля)
* Удаление файлов, связанных со старыми заявками. На данный момент этого не происходит (по аналогиями с анкетами). Но здесь явной необходимости хранить старые файлы нет.

5. API5:2023 - Нарушенная авторизация на уровне функции.

Очевидной проблемы на данный момент нет. В коде механизмов авторизации явно прописаны сущности:

* “авторизованный пользователь”, который имеет право на авторизованную работу с проектом (представитель команды, назначенной на этот проект).
* “Обычный пользователь”, который имеет доступ только к чтению публичных полей.

При этом полезными будут структурные изменения, связанные с вынесением логики авторизации на отдельный уровень, чтобы устранить возможность случайной передачи данных неавторизованным пользователям (см пункт 3. API3:2023)

6. API6:2023 - Неограниченный доступ к конфиденциальным бизнес-потокам.

Этот пункт тесно связан с проблемами, описанными в пункте 4. API4:2023.

7. API7:2023 - Подделка запроса на стороне сервера.

Единственный на данный момент очевидный случай: фильтры при поиске проектов, которые сохраняются и читаются из query-параметров в url.   
Вред этой проблемы может проявиться только в возможном появлении ошибок на сервере, не обёрнутых в обработчик ошибок, что приведёт к необходимости перезапуска сервера. Это должно будет произойти автоматически из-за использования pm2, но займёт несколько секунд. Злоумышленник тогда может без труда написать скрипт, вечно роняющий сервер.

Необходимо провести тщательное тестирование, дополнить обработчик на стороне клиента для предотвращения случайных неверных запросов и дополнить обработчик на стороне сервера для предотвращения неверных запросов от злоумышленников.

8. API8:2023 - Ошибки настроек безопасности

* Следует в явном виде задать политику CORS и CSP
* Убедиться, что у сервиса в любой момент есть действительный SSL/TLS сертификат

К этому же пункту относятся многие проблемы, затронутые в пунктах выше

9. API9:2023 - Ненадлежащее управление активами.

Может проявиться в двух видах:

* Ошибки кэширования. Несколько взаимодействующих модулей, после Breaking Change, один из которых взят из кэша, что приводит к конфликту API.  
  Не считаю проблему на данный момент актуальной. В сервисе используются современные библиотеки/фреймворки, управляющие кэшем. Реально заняться проблемой стоит только при её появлении.
* Версионирование сервиса. Не было проблемой вплоть до запуска, когда не было необходимости поддерживать сервис в постоянном функциональном состоянии.  
  На данный момент не вижу необходимости в принципиальных конкретных действиях, которые стоит заранее предпринять.

Общие рекомендации:

* Помечать breaking change в коммитах. Будет заставлять каждый раз задумываться, как изменение повлияет на другие модули системы.
* Создать граф зависимостей модулей.
* Увеличить число автоматизированных тестов.

10. API10:2023 - Небезопасное использование API.

Все касающиеся этого пункта проблемы, которые я на данный момент смог обнаружить, уже были описаны выше.