**Техническая документация Музея**

**Содержание**

[Описание классов 6](#_Toc124167203)

[Пространство имен Admin.PHP 6](#_Toc124167204)

[QueriesToPHP.cs 6](#_Toc124167205)

[HallQueries.cs 6](#_Toc124167206)

[Пространство имен Admin.Auth 8](#_Toc124167207)

[AuthPanel.cs 8](#_Toc124167208)

[VersionText.cs 8](#_Toc124167209)

[ILoggable.cs 8](#_Toc124167210)

[Logger.cs 8](#_Toc124167211)

[IFieldsProvider.cs 9](#_Toc124167212)

[LoginFieldsProvider.cs 9](#_Toc124167213)

[RegistrationFieldsProvider.cs 9](#_Toc124167214)

[PasswordSaver.cs 9](#_Toc124167215)

[Login.cs 9](#_Toc124167216)

[Registration.cs 9](#_Toc124167217)

[Пространство имен Admin.View 11](#_Toc124167218)

[HallLister.cs 11](#_Toc124167219)

[HallPreviewResizer.cs 11](#_Toc124167220)

[HallViewer.cs 11](#_Toc124167221)

[TilesDrawer.cs 12](#_Toc124167222)

[Пространство имен Admin.Edit 13](#_Toc124167223)

[ProgressSaver.cs 13](#_Toc124167224)

[EditCursor.cs 13](#_Toc124167225)

[ToolSelector.cs 13](#_Toc124167226)

[RubberBrush.cs 13](#_Toc124167227)

[PaintBrush.cs 14](#_Toc124167228)

[EditBrush.cs 14](#_Toc124167229)

[EditMedia.cs 14](#_Toc124167230)

[EditInfoBox.cs 14](#_Toc124167231)

[InfoPart.cs 14](#_Toc124167232)

[EditDecoration.cs 14](#_Toc124167233)

[HallEditor.cs 14](#_Toc124167234)

[Пространство имен Admin.New 16](#_Toc124167235)

[HallCreationDate.cs 16](#_Toc124167236)

[HallCreation.cs 16](#_Toc124167237)

[Пространство имен Admin.UsersManagement 17](#_Toc124167238)

[UsersParser.cs 17](#_Toc124167239)

[UsersSpawner.cs 17](#_Toc124167240)

[UserContent.cs 17](#_Toc124167241)

[Пространство имен Admin.Utility 18](#_Toc124167242)

[JsonHelper.cs 18](#_Toc124167243)

[RaycastUtilities.cs 18](#_Toc124167244)

[AccessLevel.cs 18](#_Toc124167245)

[User.cs 18](#_Toc124167246)

[Hall.cs 18](#_Toc124167247)

[HallContent.cs 18](#_Toc124167248)

[PanelChanger.cs 18](#_Toc124167249)

[Painting.cs 19](#_Toc124167250)

[Video.cs 19](#_Toc124167251)

[VideoFrame.cs 19](#_Toc124167252)

[Без пространства имен 20](#_Toc124167253)

[InfoController.cs 20](#_Toc124167254)

[MuseumPreviewSize.cs 20](#_Toc124167255)

[Tile.cs 20](#_Toc124167256)

[Внешние зависимости, как их заменять 21](#_Toc124167257)

[Плагин Automatic UI Anchoring 21](#_Toc124167258)

[Плагин WebGLCopyAndPaste 21](#_Toc124167259)

[Видео из GitHub Pages 21](#_Toc124167260)

[Хостинги AwardSpace и Netlify 22](#_Toc124167261)

[PHP скрипты для БД 23](#_Toc124167262)

[database\_connection.php 25](#_Toc124167263)

[delete\_content.php 25](#_Toc124167264)

[delete\_hall.php 25](#_Toc124167265)

[delete\_user.php 25](#_Toc124167266)

[get\_all\_halls.php 25](#_Toc124167267)

[get\_all\_users.php 26](#_Toc124167268)

[get\_contents\_by\_hnum.php 26](#_Toc124167269)

[get\_hall\_by\_hnum.php 26](#_Toc124167270)

[insert\_hall.php 26](#_Toc124167271)

[insert\_or\_update\_content.php 26](#_Toc124167272)

[login\_email\_count.php 26](#_Toc124167273)

[login\_full.php 27](#_Toc124167274)

[registration.php 27](#_Toc124167275)

[update\_hall.php 27](#_Toc124167276)

[update\_users.php 27](#_Toc124167277)

[Замена зависимости 27](#_Toc124167278)

[База данных MySQL 28](#_Toc124167279)

[phpMyAdmin 28](#_Toc124167280)

[Таблица users 30](#_Toc124167281)

[Таблица halls 31](#_Toc124167282)

[Таблица contents 32](#_Toc124167283)

[Замена зависимости 33](#_Toc124167284)

[Перенос проекта 35](#_Toc124167285)

[Компиляция проекта 35](#_Toc124167286)

[Версия проекта 35](#_Toc124167287)

[Компиляция 35](#_Toc124167288)

[Подготовка к релизу 36](#_Toc124167289)

[Перенос веб-приложения 37](#_Toc124167290)

[Обновление проекта на новую версию Unity 38](#_Toc124167291)

[Принятые решения и ограничения 40](#_Toc124167292)

[Почему MySQL? 40](#_Toc124167293)

[Почему PHP? 41](#_Toc124167294)

[Почему хостинг AwardSpace и Netlify? 41](#_Toc124167295)

[AwardSpace 41](#_Toc124167296)

[Netlify 41](#_Toc124167297)

[Почему видео из GitHub Pages? 41](#_Toc124167298)

# Описание классов

## Пространство имен Admin.PHP

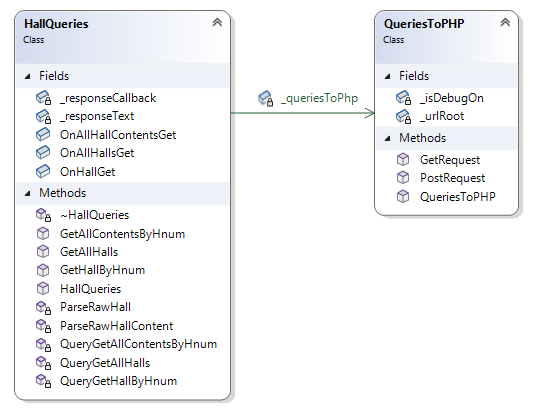


Рис. 1. Диаграмма классов

### QueriesToPHP.cs

Класс, отвечающий за GET/POST запросы к PHP скриптам. Запросы выполняются внутри IEnumerator методах (корутинах), которые принимают название PHP скрипта на сервере, возвращают – обратный вызов с текстом-ответом из скрипта. Для использования класса нужно сконструировать его C# объект, обращаться к методам через объект.

### HallQueries.cs

Класс, отвечающий за ООП запросы к базе данных, относящиеся к залам музея. Имеет 3 публичных контракта-корутины – GetHallByHnum (получить зал по его ID), GetAllHalls (получить все залы), GetAllContentsByHnum (получить все наполнение зала по его ID). При успешном выполнении этих методов, вызываются обратные вызовы OnHallGet, OnAllHallsGet, OnAllHallContentsGet, которые передают объекты или списки объектов. Для использования класса нужно сконструировать его C# объект, обращаться к методам через объект.

## Пространство имен Admin.Auth

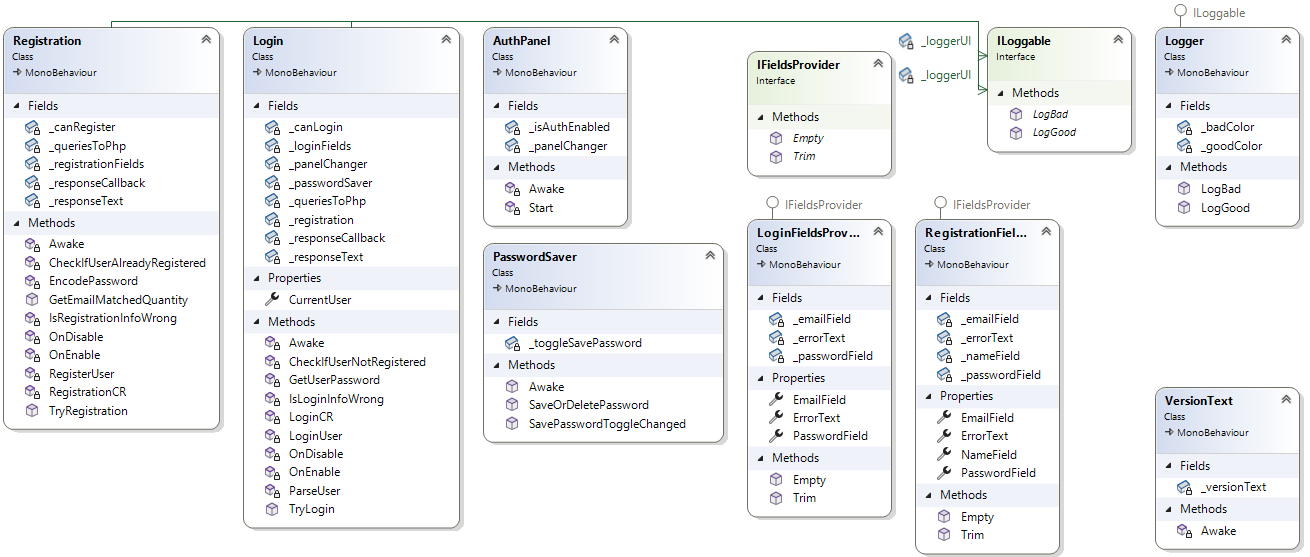


Рис. 2. Диаграмма классов

### AuthPanel.cs

MonoBehaviour, который устанавливает полноэкранный режим на старте, а также включает панель авторизации (либо пропускает ее при выключенном флаге \_isAuthEnabled).

### VersionText.cs

MonoBehaviour, который на старте приложения отображает версию приложения в виде текста в интерфейсе.

### ILoggable.cs

Интерфейс, содержащий методы LogGood, LogBad для логгирования хорошего или плохого сообщения.

### Logger.cs

MonoBehaviour, заключен контракт с интерфейсом ILoggable, дает в использование публичные методы для логгирования в пользовательском интерфейсе.

### IFieldsProvider.cs

Интерфейс, содержащий методы Empty (очистить содержимое поля), Trim (обрезать поле с двух сторон).

### LoginFieldsProvider.cs

MonoBehaviour, заключен контракт с интерфейсом IFieldsProvider, дает в использование публичные методы для манипуляции с полями ввода текста, используемые при логине пользователя.

### RegistrationFieldsProvider.cs

MonoBehaviour, заключен контракт с интерфейсом IFieldsProvider, дает в использование публичные методы для манипуляции с полями ввода текста, используемые при регистрации пользователя.

### PasswordSaver.cs

MonoBehaviour, который занимается логикой сохранения или удаления пароля, если отмечена галочка «Запомнить пароль». Использует PlayerPrefs для сохранения (максимальный вес сохраняемых данных – 1МБ, сохраняются они в IndexedDB браузера).

### Login.cs

MonoBehaviour, который занимается логикой авторизации пользователя, также получает пользователя из базы данных и запоминает его в публичное авто-свойство CurrentUser, которое используется множеством других классов.

### Registration.cs

MonoBehaviour, который занимается логикой регистрации пользователя.

## Пространство имен Admin.View

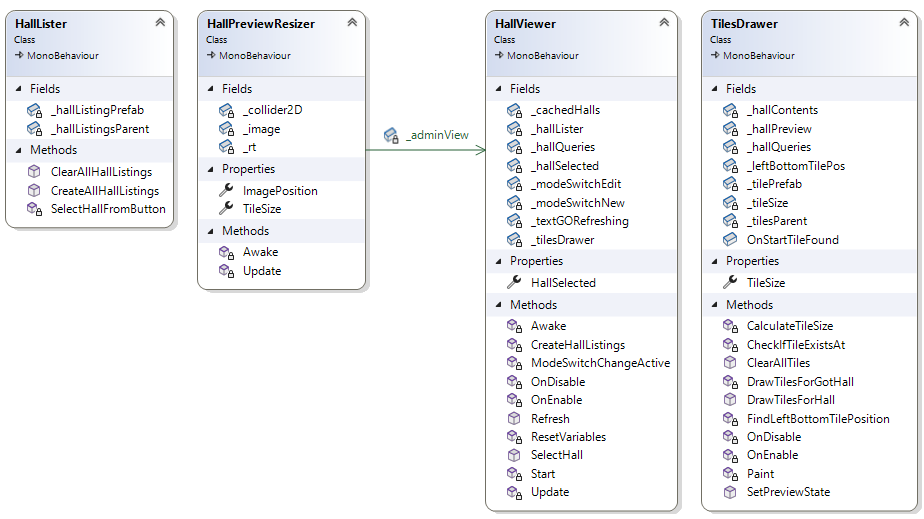


Рис. 3. Диаграмма классов

### HallLister.cs

MonoBehaviour, отвечающий за создание и удаление интерфейсных элементов – списка доступных залов.

### HallPreviewResizer.cs

MonoBehaviour, отвечающий за изменение размера предварительного просмотра у выбранного 2D плана зала.

### HallViewer.cs

MonoBehaviour, отвечающий за выбор зала. Хранит в себе публичное свойство HallSelected, которое используется множеством других классов.

### TilesDrawer.cs

MonoBehaviour, отвечающий за получение содержимого выбранного зала, его отрисовку в интерфейсе, поверх 2D плана.

## Пространство имен Admin.Edit

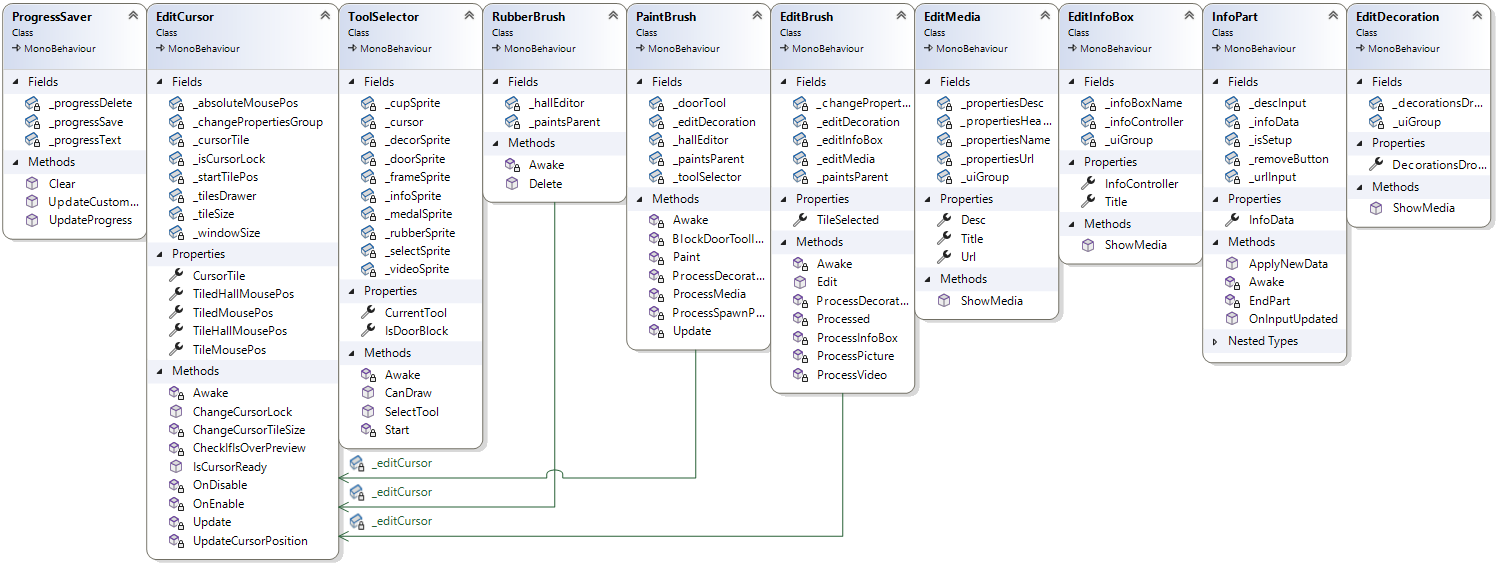


Рис. 4. Диаграмма классов

### ProgressSaver.cs

MonoBehaviour, отвечающий за визуальное отображение в тексте прогресса сохранения настроек зала.

### EditCursor.cs

MonoBehaviour, отвечающий за просчет положения и отображение курсора в интерфейсе.

### ToolSelector.cs

MonoBehaviour, отвечающий за выбор инструмента из палитры, а также изменение иконки курсора.

### RubberBrush.cs

MonoBehaviour, отвечающий за удаление иконки на месте курсора при выбранном инструменте резинки.

### PaintBrush.cs

MonoBehaviour, отвечающий за рисование выбранной иконки из палитры на 2D плане зала, а также установление начального состояния нарисованного предмета музея.

### EditBrush.cs

MonoBehaviour, отвечающий за редактирование (изменение) предмета на месте курсора, его обработка в зависимости от его типа.

### EditMedia.cs

MonoBehaviour, отвечающий за отображение настроек медиа (фото, видео) на пользовательском интерфейсе.

### EditInfoBox.cs

MonoBehaviour, отвечающий за отображение настроек информационного стенда на пользовательском интерфейсе.

### InfoPart.cs

MonoBehaviour, отвечающий за часть информации для информационного стенда – проверка ввода, сохранение, удаление. Также содержит структуру InfoPartData, описывающую ссылка на картинку и описание.

### EditDecoration.cs

MonoBehaviour, отвечающий за отображение настроек декорации на пользовательском интерфейсе.

### HallEditor.cs

MonoBehaviour, отвечающий за редактирование настроек зала – даты открытия/закрытие, название зала, его доступность, удаление. Также хранит в себе массив, являющийся 2D планом содержимого зала. В нем происходят все взаимодействия с базой данных – сохранение настроек зала, содержимого, их удаление.

## Пространство имен Admin.New

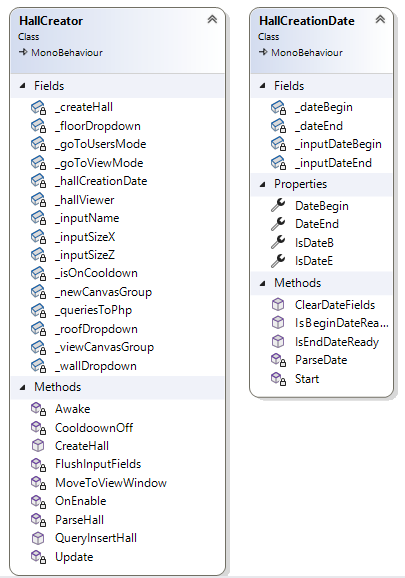


Рис. 5. Диаграмма классов

### HallCreationDate.cs

MonoBehaviour, отвечающий за ввод даты открытия и закрытия зала пользователем.

### HallCreation.cs

MonoBehaviour, отвечающий за парсинг и валидацию введенных данных нового зала, отправляет запрос к базе данных на его создание.

## Пространство имен Admin.UsersManagement

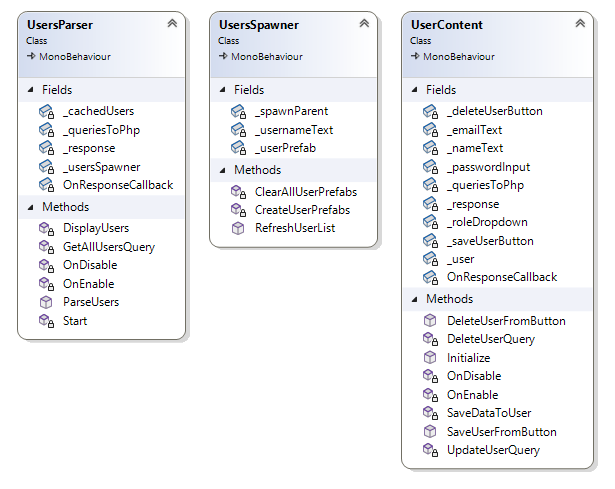


Рис. 6. Диаграмма классов

### UsersParser.cs

MonoBehaviour, отвечающий за получение всех пользователей из БД, кеширует их в список пользователей.

### UsersSpawner.cs

MonoBehaviour, отвечающий за создание всех кешированных пользователей на пользовательском интерфейсе.

### UserContent.cs

MonoBehaviour, отвечающий за представление пользователя в интерфейсе – отображение его полей, сохранение изменений у пользователя в базе данных и удаление пользователя.

## Пространство имен Admin.Utility

### JsonHelper.cs

Класс, отвечающий за сериализацию и десериализацию массивов в JSON и обратно. Доступный по умолчанию в Unity класс JsonUtility не способен корректно это делать.

### RaycastUtilities.cs

Класс, отвечающий за получение элемента интерфейса, на который указывает курсор.

### AccessLevel.cs

Перечисление, описывающее роли, возможные у пользователя.

### User.cs

Структура данных, описывающая пользователя. Содержит в себе AccessLevel.

### Hall.cs

Структура данных, описывающая зал Музея, совпадает с структурой соответствующей таблицы в базе данных.

### HallContent.cs

Структура данных, описывающая наполнение зала Музея, совпадает с структурой соответствующей таблицы в базе данных.

### PanelChanger.cs

Содержит перечисление Panel, отвечающее за UI-панели в приложении, а также логику переключения между этими панелями.

### Painting.cs

MonoBehaviour, отвечающий за загрузку и отображение картинки из Интернета по ссылке.

### Video.cs

MonoBehaviour, отвечающий за загрузку и воспроизведение видео в текстуре из Интернета (GitHub Pages) по ссылке.

### VideoFrame.cs

MonoBehaviour, отвечающий за загрузку и воспроизведение видео из Интернета (GitHub Pages) по ссылке при взаимодействии с пользователем.

## Без пространства имен

### InfoController.cs

MonoBehaviour, отвечающий за управление информационным стендом: добавление и удаление его частей, сериализация всей информации в JSON для последующего сохранения в БД.

### MuseumPreviewSize.cs

MonoBehaviour, отвечающий за изменение размера предварительного просмотра плана 2D музея.

### Tile.cs

MonoBehaviour, отвечающий за визуальное представление иконки на 2D плане Музея.

# Внешние зависимости, как их заменять

## Плагин Automatic UI Anchoring

[Unity Asset Store](https://assetstore.unity.com/packages/tools/gui/automatic-ui-anchoring-142911)

При наличии папки в проекте позволяет автоматически расставлять якоря на элементах пользовательского интерфейса нажатием F1 с выбранным элементом. Стабилен и никак не влияет на работу проекта, Unity редактора. Не является обязательным. Можно просто удалить папку плагина из проекта. Можно не использовать. Нужен исключительно для удобства расстановки якорей.

## Плагин WebGLCopyAndPaste

[GitHub](https://github.com/greggman/unity-webgl-copy-and-paste)

При наличии папки в проекте позволяет пользователям в WebGl приложении в браузере вставлять текст в поля ввода (TMP\_InputField) извне. Без этого плагина вставка текста работает только, если текст был скопирован в самом приложении. Пока показывает себя стабильным. Является обязательным, так как вставка текста извне важна для редакторов. У него есть один аналог – [WebGlInput](https://github.com/kou-yeung/WebGLInput), но он был протестирован и оказался не рабочим для проектов на Unity версии 2021 и выше, не обновлялся. Поэтому, этот плагин является необходимым и незаменимым.

## Видео из GitHub Pages

[Основной репозиторий](https://github.com/MaratG2/istumuseumstorage) хранилища для видео.

Подробнее про процесс загрузки и использования видео из GitHub Pages описан в инструкции по администрированию Музея, в пункте «Настройка – видео». Почему именно используется такая система описана в главе «Принятые решения и ограничения».

Проект Музея зависит лишь от ссылки на видео, самому проекту не важно, где оно расположено. Соответственно, для переноса нужно лишь загрузить видео на другой сервис, отличный от GitHub Pages. Но важно, чтобы это не нарушало CORS-политику, иначе видео воспроизводится не будет, возникнет ошибка.

Возможное место для улучшения – хранить видео на том же сервере (домене), где находится веб-приложение Музея – тогда обращение из доменного имени к самому себе не будет нарушать CORS-политику и видео будет нормально загружаться, без «танцев с бубнами» с GitHub Pages.

## Хостинги AwardSpace и Netlify

На Netlify хостятся HTML5 WebGl приложения административной и клиентской части Музея. В этих целях можно использовать также, например, Heroku, или мощности ИжГТУ. Важно то, что на Netlify есть SSL сертификат (HTTPS), а на AwardSpace его нет в бесплатном доступе. Поэтому, чтобы в приложении не возникало ошибок безопасности при обмене информацией между HTTPS и HTTP ресурсами, используется проксирование (файл \_redirects), который маскирует Netlify как HTTP, из-за чего обмен информацией происходит между HTTP и HTTP, что допустимо и не вызывает ошибок.

На AwardSpace лежат PHP скрипты для работы с базой данных (причем это не просто файловое хранилище, в AwardSpace есть runtime выполнение PHP), а также хранится MySQL база данных.

Данные аккаунта для входа в хостинг AwardSpace, как и ссылка на хостинг, находятся в папке проекта, в папке Database\AwardSpace.

Подробнее про перенос веб-приложения в главе «Перенос проекта», а про PHP и БД – в частях ниже. Если будете переносить PHP или БД, то не переносите что-то одно, переносите их вместе, и берите во внимание их особенности.

## PHP скрипты для БД

Локальные PHP скрипты хранятся в папке проекта, в папке Database\PHP. На [AwardSpace](https://www.awardspace.com/user-login/) сервере их можно найти и загрузить в следующем месте:

Войдите в панель AwardSpace, промотайте страницу вниз и нажмите на File Manager:

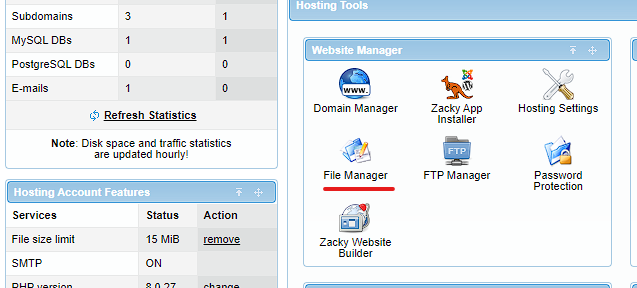


Рис. 7.

Перейдите в папку museumistu.atwebpages.com/→PHP/. В ней лежат PHP скрипты, к которым обращаются из кода Музея. Чтобы загрузить новый, нажмите на кнопку Upload и перетащите PHP файлы в появившееся окно. Кнопка находится как показано на рисунке:

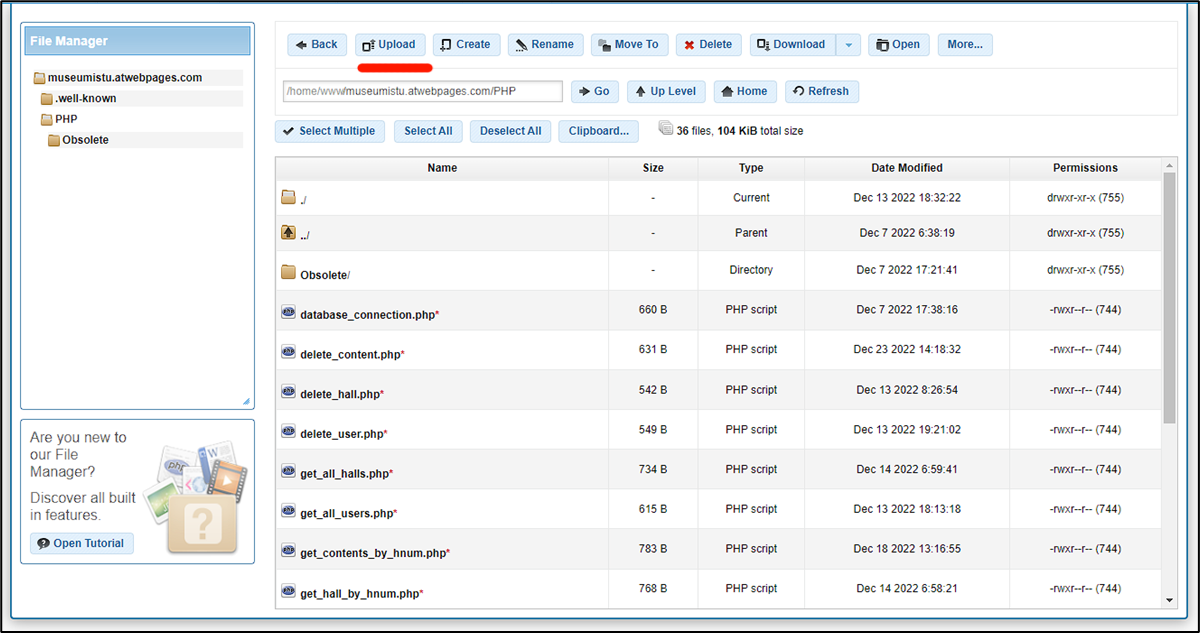


Рис. 8.

После загрузки PHP скрипта, его можно будет выполнять по ссылке, например подключение к базе данных находится по ссылке:

<http://museumistu.atwebpages.com/PHP/database_connection.php>

Важно, чтобы в каждом PHP скрипты были следующие заголовки, чтобы их выполнение было разрешено CORS-политикой браузеров:

header("Access-Control-Allow-Credentials: true");

header('Access-Control-Allow-Origin: \*');

header('Access-Control-Allow-Methods: POST, GET, OPTIONS');

header('Access-Control-Allow-Headers: Accept, X-Access-Token, X-Application-Name, X-Request-Sent-Time');

Далее приведено описание всех использующихся PHP скриптов:

### database\_connection.php

Содержит функцию подключения к базе данных, которое возвращает это подключение вызывающей стороне – для дальнейшего использования. Каждый другой PHP скрипт включает в себя database\_connection.php, кеширует себе подключение.

**Важно –** данные для подключения к базе данных содержатся в строковых константах в файле.

### delete\_content.php

Удаляет предмет зала Музея по номеру зала, из которого нужно удалить, а также по позиции самого предмета.

### delete\_hall.php

Удаляет зал Музея по его номеру.

### delete\_user.php

Удаляет зарегистрированного в администраторской части Музея пользователя по его адресу электронной почты.

### get\_all\_halls.php

Возвращает все записи таблицы залов Музея по возрастанию их ID. Формат возвращаемых данных: string. Cтолбцы записи разделены между собой символом '|', записи между собой – символом ';'.

### get\_all\_users.php

Возвращает все записи таблицы пользователей, зарегистрированных в администраторской части Музея. Формат возвращаемых данных: string. Cтолбцы записи разделены между собой символом '|', записи между собой – символом ';'.

### get\_contents\_by\_hnum.php

Возвращает все записи таблицы предметов зала, чей номер зала совпадает с передаваемым в качестве аргумента. Формат возвращаемых данных: string. Cтолбцы записи разделены между собой символом '|', записи между собой – символом ';'.

### get\_hall\_by\_hnum.php

Возвращает запись таблицы залов Музея по его уникальному номеру. Формат возвращаемых данных: string. Cтолбцы записи разделены между собой символом '|'.

### insert\_hall.php

Добавляет новую запись в таблицу залов Музея с передаваемыми настройками.

### insert\_or\_update\_content.php

Добавляет новую или обновляет существующую запись в таблицу предметов зала. Обновление происходит, если передаваемая позиция объекта уже существует (выполняется ON DUPLICATE KEY).

### login\_email\_count.php

Возвращает количество записей в таблице пользователей с передаваемым адресом электронной почты. Используется в двух местах: проверить, есть ли такой пользователь при логине и регистрации. Формат возвращаемых данных: string (int).

### login\_full.php

Возвращает запись из таблицы пользователей по передаваемому адресу электронной почты и паролю. Формат возвращаемых данных: string. Cтолбцы записи разделены между собой символом '|', записи между собой – символом ';'.

### registration.php

Добавляет новую запись в таблицу пользователей с передаваемыми данными.

### update\_hall.php

Обновляет настройки зала по его уникальному номеру.

### update\_users.php

Обновляет пользователя по его почте.

### Замена зависимости

В текущий момент, PHP-скрипты и БД очень сильно связаны и зависимы.

Во-первых, скрипты написаны для MySQL БД. Если БД будет, например, PostgreSQL, нужно будет переписывать скрипты. Но изменение из MySQL в PostgreSQL крайне не рекомендуется, потому что были проблемы с подключением к БД из PHP скрипта, SSL-сертификатами.

Во-вторых, PHP-скрипты и БД должны быть на одном хостинге, домене, сервере. Если вы оставите БД на AwardSpace, а PHP-скрипты перенесете на другой сервер, то они перестанут работать из-за нарушения CORS-политики.

В-третьих, ссылка на корень сервера, где хранятся PHP-скрипты, прописана в файле \_redirects папки Databases, этот файл затем кладется во все сборки приложения, если используется связка AwardSpace+Netlify. Если PHP-скрипты хранятся на другом сервере, а веб-приложение остается на Netlify, то нужно заменить ссылку в файле \_redirects.

В-четвертых, ссылка на корень сервера, где хранятся PHP-скрипты, прописана в скрипте Музея «QueriesToPHP.cs». Если вы перенесли хостинг PHP-скриптов, то измените поле \_urlRoot на ссылку на корневую папку со всеми скриптами.

Также, PHP-скрипты обязательно должны храниться на хостинге, который поддерживает их runtime выполнение. То есть, просто хранить их на Google Drive или Netlify не получится.

## База данных MySQL

### phpMyAdmin

Попасть в phpMyAdmin (управляющую панель базы данных) можно в панели AwardSpace:

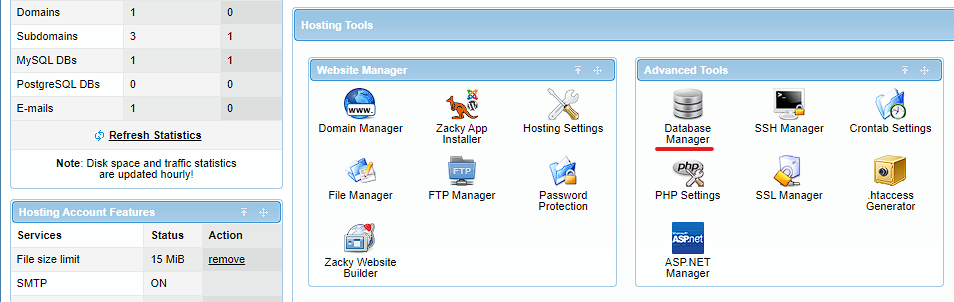


Рис. 9.

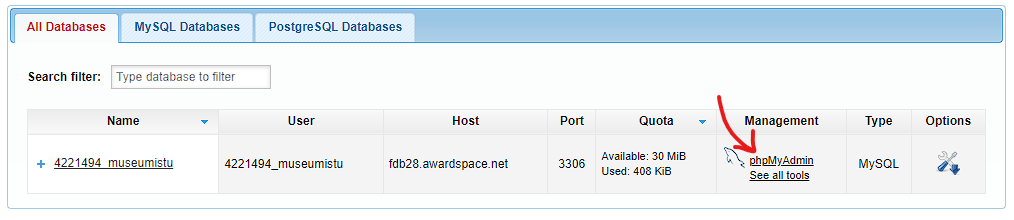


Рис. 10.

Если нажать не на phpMyAdmin, а на See all tools – откроется список инструментов, где можно, например, импортировать или экспортировать базу данных, чтобы, например, перенести базу данных на другой хостинг:

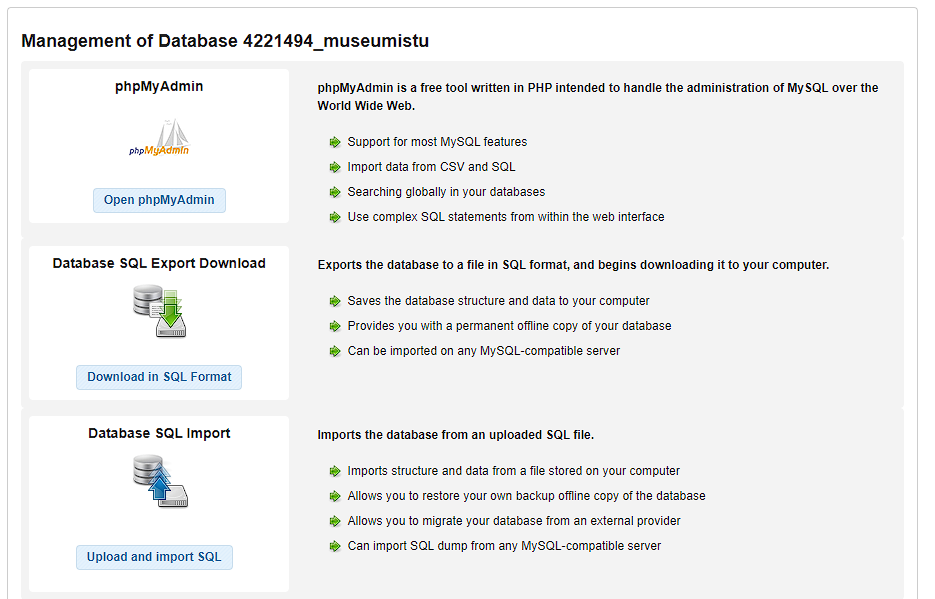


Рис. 11.

Как работать с панелью администрирования phpMyAdmin подробно описано в [текстовом гайде](https://losst.pro/kak-polzovatsya-phpmyadmin).

Далее приведено описание всех текущих таблиц Музея:

Таблица users (пользователи):

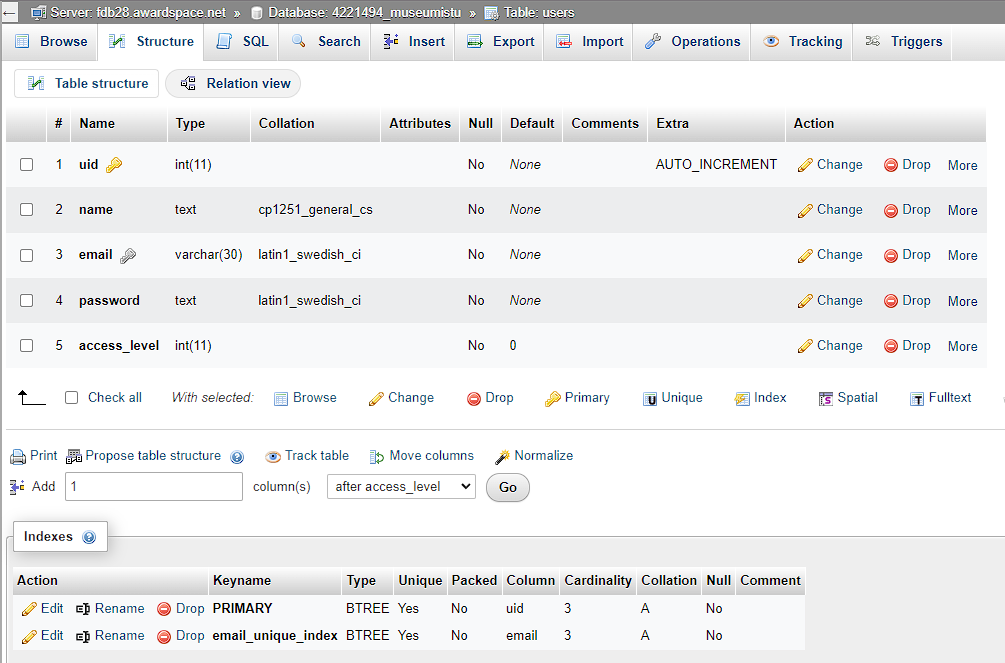


Рис. 12.

Таблица halls (залы Музея):

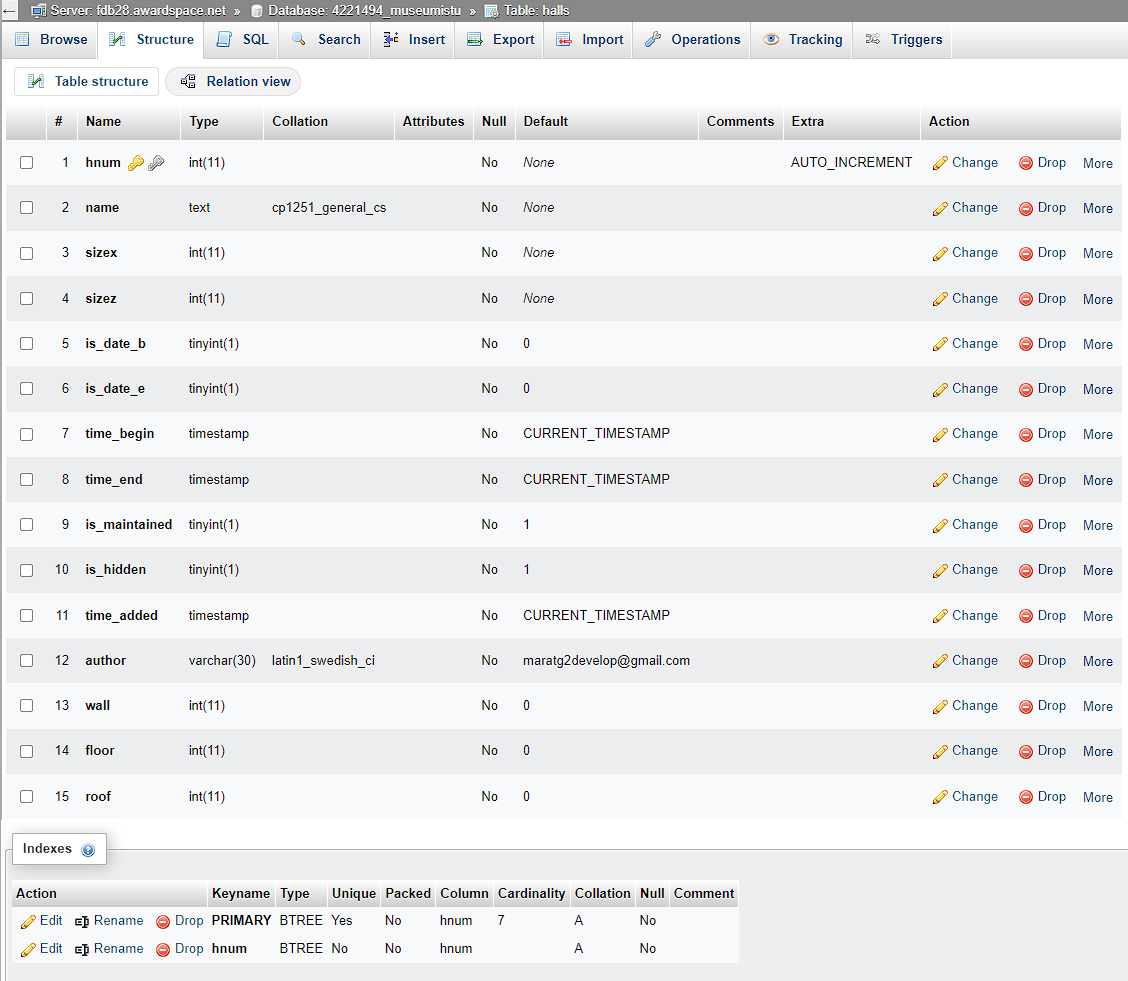


Рис. 13.

Таблица contents (наполнение зала):

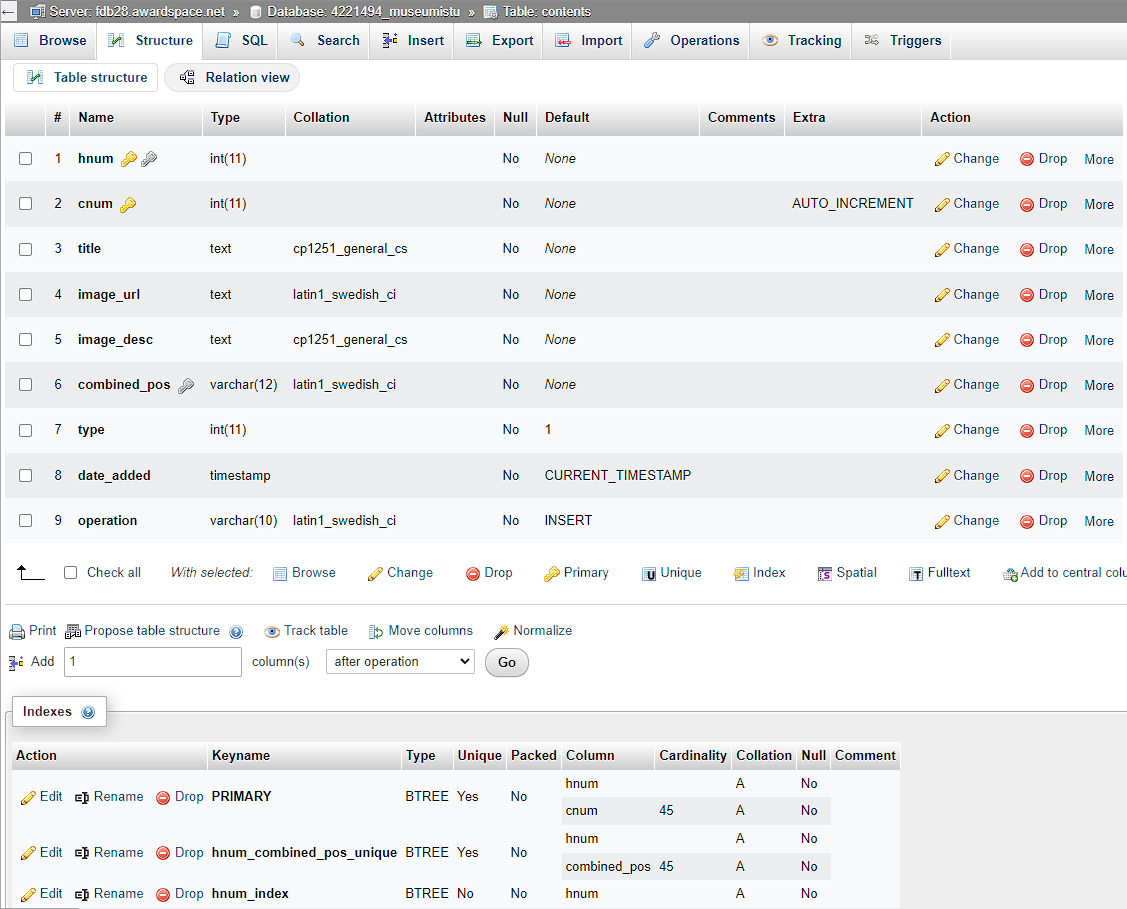


Рис. 14.

### Замена зависимости

В текущий момент, PHP-скрипты и БД очень сильно связаны и зависимы.

Во-первых, скрипты написаны для MySQL БД. Если БД будет, например, PostgreSQL, нужно будет переписывать скрипты. Но изменение из MySQL в PostgreSQL крайне не рекомендуется, потому что были проблемы с подключением к БД из PHP скрипта, SSL-сертификатами.

Во-вторых, PHP-скрипты и БД должны быть на одном хостинге, домене, сервере. Если вы оставите PHP-скрипты на AwardSpace, а БД перенесете на другой сервер, то PHP-скрипты перестанут работать из-за нарушения CORS-политики.

Чтобы перенести БД, достаточно ее экспортировать с помощью инструментов администрирования AwardSpace. Затем в ином хостинге, с помощью инструментов администрирования, импортировать базу. Далее, в PHP-скрипте database\_connection.php нужно изменить данные для подключения к БД.

# Перенос проекта

## Компиляция проекта

### Версия проекта

Сперва зайдите в настройки проекта и обновите номер версии в вкладке Player, поле Version:

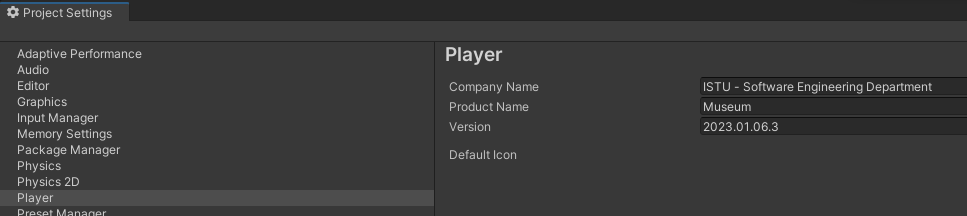


Рис. 15.

Рекомендуем придерживаться одного стиля наименования версии: yyyy.MM.dd.X, где X – номер версии, начиная с 1, по возрастанию, сбрасывается каждый день. Например, вторая компиляция 31 декабря 2021 будет под версией 2021.12.31.2.

### Компиляция

Перейдите в Build Settings (ctrl+shift+b). Во-первых, убедитесь, что выбрана именно WebGL платформа для компиляции (рис. 16.1). Во-вторых, убедитесь, что в сборке выбраны нужные сцены – для панели администратора это сцена Admin/Scenes/AdminPanel (рис. 16.2). В-третьих, нажмите на кнопку Build (рис. 16.3), в выборе пути сборки перейдите в папку Builds проекта (если нет – создайте), создайте там папку с названием версии (например, «MuseumAdmin\_2023.01.06.3»), выберите ее и начните компиляцию.

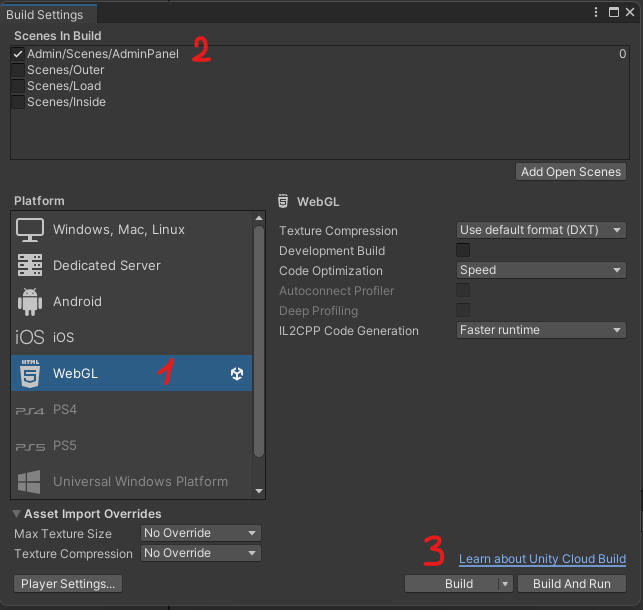


Рис. 16.

### Подготовка к релизу

При текущем порядке работы, когда PHP-скрипты и БД хранятся на AwardSpace (HTTP, без SSL сертификата), а веб-приложение на Netlify (HTTPS, с сертификатом), без дополнительной настройки приложение будет выбрасывать ошибки. Ошибки связаны с безопасностью, так как безопасное HTTPS приложение пытается обмениваться информацией с небезопасным HTTP ресурсом.

Для решения данной проблемы используется проксирование внутри Netlify с помощью файла “\_redirects”. Его содержимое – одна строчка, но этот файл применим только для хостинга веб-приложения на Netlify:

/api/\* http://museumistu.atwebpages.com/:splat 200

Этот файл маскирует запросы из Netlify как из HTTP, из-за чего обмен информацией происходит между HTTP и HTTP, что допустимо и не вызывает ошибок.

**То есть, важно –** возьмите файл \_redirects из папки Database проекта,скопируйте его внутрь папки с сборкой (там, где лежит html файл). Получится следующее:

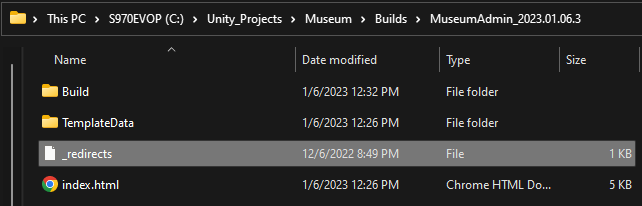


Рис. 17.

Теперь релизная сборка готова для публикации в Интернете, например в Netlify.

## Перенос веб-приложения

Чтобы перенести веб-приложение из хостинга Netlify на иной, нужно решить одну проблему. Просто загрузки приложения на другой хостинг (например, загрузить папку на Heroku) – будет недостаточно.

Во-первых, нужно, чтобы сервис поддерживал работу HTML5 WebGL приложение (например, загрузив веб-приложение на Google Drive, оно не будет запускаться и работать).

Во-вторых, нужно решить проблему с взаимодействием между PHP+БД и веб-приложением, не нарушая политику безопасности. Как было описано ранее, нужно, чтобы либо оба они были HTTP, либо оба – HTTPS. Если веб-приложение HTTPS, а PHP+БД хранятся на HTTP, то у веб-приложения нужно делать проксирование, которое зависит от сервиса. Текущее решение работает только под Netlify.

Если эти проблемы решены, то достаточно просто загрузить веб-приложение на хостинг – PHP-скрипты и база данных будут работать.

## Обновление проекта на новую версию Unity

В нормальных условиях проект обновлять на новую версию Unity не нужно. На момент написания документации используется версия 2021.3.7f1. Даже если у вас нет этой версии, а есть новее – обновлять проект не следует из-за возможных багов (несовместимости, …). Скачать эту версию Unity можно из официального архива, или [по ссылке](https://unity.com/releases/editor/whats-new/2021.3.7).

Если же по какой-то причине нужно обновлять версию, например, по утрате поддержки текущей версии, то советуем переходить на последнюю доступную версию LTS (Long Term Support, длительная поддержка версии компанией Unity).

Unity значительно меняют свой игровой движок каждый год, так что если предыдущая версия была, например, 2021 года, а новая – 2022, то, возможно, появятся некоторые баги – например, устареет какая-то часть кода и ее нужно будет переписывать на новый API.

# Принятые решения и ограничения

## Почему MySQL?

В начале проекта был выбран PostgreSQL с крайне удобной C# библиотекой Npgsql. К сожалению, в WebGL приложении с Unity невозможно использовать данную библиотеку – это связано опять с нарушением безопасности. Библиотека работает через WebSockets, которые запрещены в HTML5 WebGL, при попытке обратиться к БД возникала ошибка:

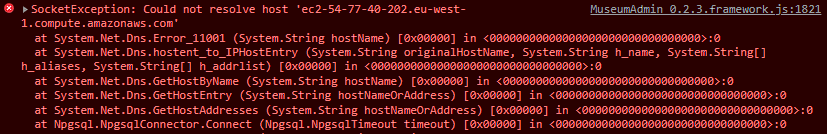


Рис. 18.

Самим писать C# обертку для Npgsql – слишком дорого, поэтому мы отказались от этой библиотеки. Следующий вариант – использовать PHP-скрипты для подключения к PostgreSQL.

Возникала проблема с подключением из-за настроек безопасности СУБД в бесплатных сервисах, где хранилась база данных (например, Heroku) – из-за того, что они бесплатные, настройки нельзя было менять. В настройках по умолчанию нет доступа к подключению к БД с любого IP – поэтому от PostgreSQL полностью отказались.

Подключение к MySQL в PHP-скриптах через mysqli\_connect не вызывало никаких проблем – тем более, в хостинге, где хранятся PHP-скрипты, доступна одна бесплатная MySQL база данных. Взаимодействие между PHP и БД на одном хостинге, домене – нет нарушений безопасности.

## Почему PHP?

Так как библиотеки-обертки баз данных для Unity в WebGL не работают, нужно использовать другое решение для взаимодействия с базой данных. PHP достаточно прост и удобен в использовании, хотя у него есть свои ограничения. Также с PHP был предыдущий опыт, с ним приложение функционирует как и ожидается.

Из иных решений, можно написать C# сервер, который будет служить прослойкой между клиентом Музея и базой данных (сейчас, можно сказать, у нас PHP-сервер). Это решение наверняка эффективнее справляется со своей работой, но оно значительно дороже.

## Почему хостинг AwardSpace и Netlify?

### AwardSpace

Было перебрано большинство отечественных и зарубежных онлайн хостингов. AwardSpace имеет бесплатный домен, runtime выполнение PHP-скриптов, бесплатную MySQL базу данных. Буквально единственный найденный такой вариант из бесплатных.

### Netlify

Удобный и простой в использовании хостинг HTML5 приложений, отвечающий практически всем нуждам (следующий шаг, наверное, хостинг на мощностях ИжГТУ).

## Почему видео из GitHub Pages?

Проблема возникает в CORS механизме браузеров (Cross-Origin Resource Sharing), если редактор Музея будет вставлять ссылки на видео из YouTube, ВКонтакте, Google Drive, Yandex Disk, то видео не будет воспроизведено из-за CORS-ошибки.

Было проведено исследование решений – перечисленные выше сервисы не работают. Плагины, например [UnityYoutubePlayer](https://github.com/iBicha/UnityYoutubePlayer) – тоже. Единственный найденный рабочий вариант, не нарушающий CORS-политику – ссылка на видео из GitHub Pages.

Пример CORS ошибки (здесь использовался плагин UnityYoutubePlayer):

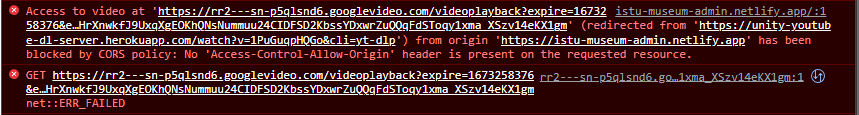


Рис. 19.