ВВЕДЕНИЕ

Вас пригласили в команду разработки нового амбициозного проекта - “Умный город”. На первой итерации было решено разработать систему информирования коммунальных служб о проблемах в городе на примере несанкционированных мусорных свалок и дорожных ям. Команда уже разработала API для работы с базой данных системы. API позволяет просматривать существующие проблемы, производить регистрацию пользователей. Для зарегистрированных пользователей существует возможность оставлять заявки с описанием проблем, редактировать и скрывать свои заявки, писать комментарии и редактировать.

Ваша задача - разработать мобильное приложение для ОС Android, который использует этот API.

ОПИСАНИЕ ПРОЕКТА И ЗАДАЧИ

В рамках системы под заявкой понимается описание какой то проблемы (например яма на дороге или несанкционированная свалка мусора). В системе проблемы разделены на типы, заявка может принадлежать только одному из типов.

Информация о заявке содержит следующие параметры:

1. Заголовок
2. Дата регистрации заявки
3. Тип заявки
4. Текстовое описание
5. Местоположение (текст)
6. GPS координаты - широта и долгота
7. Фотография заявки
8. Автор заявки
9. Тип заявки

Тип заявки - объект со следующими параметрами:

1. Название типа
2. Описание типа
3. Тег

На данный момент в системе доступно два типа заявок (название/тег): Ямы (pit), Несанкционированный мусор (trash).

Приложение может работать в двух режимах - авторизованном и не авторизованном. Изначально (при первом запуске) приложение сразу загружается в неавторизованном режиме. В этом режиме должна быть доступна часть возможностей, которую мы назовем “публичной”. Одной из таких возможностей является авторизация на сервере (через API), после чего становятся доступны дополнительные возможности (будем называть “их закрытыми”).

Публичные возможности

На главном экране должно отображаться 5 последних (по дате подачи) заявок. Заявки отображаются в виде карточек с фотографией, заголовком заявки и датой публикации. При нажатии на карточку нужно отобразить экран с подробной информацией о заявке.

На экране заявки необходимо вывести всю информацию о заявке, а также комментарии пользователей к этой заявке. В случае если пользователь авторизован, должна быть возможность добавить комментарий.

Также должен быть доступен экран авторизации (для неавторизованных пользователей).

Дополнительные задания:

На главном экране, в случае, если заявок больше необходимо предусмотреть прокрутку карточек заявок. Также нужно предусмотреть возможность фильтрации заявок по типам.

Для авторизованных пользователей необходимо добавить возможность изменять свои комментарии. Также при отображении комментариев для авторизованных выделить их (например цветом).

На экране заявки встроить Яндекс/гугл/2 ГИС карту на которой отображать точку по координатам (GPS). Как вариант предложить дополнительные сервисы связанные с GPS, например отображение населенного пункта/улицы.

Закрытые возможности

Возможности должны быть доступна после ввода логина и пароля на экране авторизации. Авторизация должна сохранятся после переходов между экранами. При этом должна быть предусмотрена функция выхода.

После успешного ввода логина и пароля пользователю должен быть доступен экран “Личный кабинет”, на которой отображаются данные пользователя.

Также для авторизованных пользователей необходимо реализовать два экрана:

- Экран “Мои заявки”, на котором нужно отобразить все заявки оставленные пользователем. На экране нужно разместить быть кнопку “Добавить”. Также вывести заявки пользователя. Заявки необходимо упорядочить по дате публикации. Должна быть реализована возможность редактирования заявок.
- Экран “Мои комментарии”, на экране нужно отобразить все комментарии пользователя. Должна быть возможность редактирования комментариев.

Экраны описанные выше не должны быть доступны для неавторизованного пользователя.

Примечание: Пока API не поддерживает эту функцию загрузки изображений! При создании и редактировании заявок в качестве имени файла отправляйте stop.png (см описание API).

Описание API

API для сервиса размещается по общему адресу: https://webcomp.bsu.ru/api/finals24

Ниже описаны возможности, которые предоставляет API, для сокращения описания приведена ссылка на конкретную функцию. Полная ссылка формируется в результате конкатенации (сложения) общего адреса и ссылки (например: https://webcomp.bsu.ru/api/finals24/reports/all).

Все запросы представлены в архиве, папка comments - запросы для комментариев, папка reports - для заявок. Примеры с публичными запросами в названии файла содержат слово public. Авторизованные запросы имеют одинаковую логику в именовании (по содержанию ключевого слова в имени файла): add - добавление сущности, delete - удаление сущности, update - обновление сущности.

Примечание: Изображения возвращаются как имена файлов.

Все файлы располагаются по адресу: https://webcomp.bsu.ru/uploads/itbur2024

Пример: https://webcomp.bsu.ru/uploads/itbur2024/img_2.jpg

При загрузке собственных заявок можно указывать изображение по умолчанию: stop.png

Для тестов можно скачать файлы по ссылке:
https://drive.google.com/file/d/1rjx-pNtCtQwaHqMW2uT7t7YkhmsTiVEX/view?usp=sharing

Также доступны данные заявок:

https://docs.google.com/spreadsheets/d/1nTznbP5ZZVdRg-9kHSTKsOomQECdBmUTEPBHk5hJduQ/edit?usp=sharing

Архив картинок:
https://drive.google.com/file/d/1xb6JU49bViW3jBamz12IYlzd0GsQGEbY/view?usp=sharing

Публичные запросы

Публичные запросы доступны всем пользователям в сети Интернет и могут быть отправлены из браузера.

Список заявок

GET (reports/all) - список заявок. Система возвращает массив объектов (все заявки)

Request	**Response**

Content-type: application/json	**Success:**
	В базе данных существует проект
	Status: 200/OK
	Content-type: application/json
	Body:
	“data”:[
	{
	"id": 1,
	"title": "Яма на Ранжурова",
	"content": "Очень большая яма, не меньше 50 сантиметров!",
	"location": "Рядом со второй школой",
	"latitude": 51.8357841,
	"longitude": 107.5389092,
	"img_link": "img_1.jpg",
	"author": {
	"team_name": "Организатор трека Vue.js",
	"description": "Тут моя краткая биография2",
	"track_id": 2,
	"track_name": "Фронтенд разработка на Vue.js ",
	"track_description": "Участники трека…. ",
	"user_name": "Хабитуев Баир Викторович",
	"updated_at": "2024-03-08T03:07:07.000000Z"
	},

	"report_type": {
	"id": 1,
	"created_at": "2024-03-21T12:29:55.000000Z",
	"updated_at": "2024-03-21T12:29:55.000000Z",
	"title": "Ямы",
	"description": "Ямы города Улан-Удэ",
	"tag": "pit"
	}
	},
	
	]
	Denied
	проблемы сервиса
	Status: 404

Информация о заявке

GET (reports/{id}) - информация о заявке. Система принимает на вход идентификатор проекта и возвращает информацию об одной заявке.

Пример ссылки (для проекта с id=3): https://webcomp.bsu.ru/api/finals24/reports/3

Request	**Response**

Content-type: application/json
 	
Parameters:
 	
в качестве параметра в строке передается id заявки.
	Success:
	
	В базе данных существует проект
	
	Status: 200/OK
	
	Content-type: application/json
	
	Body: "data": {
	
	"id": 3,
	
	"title": "Яма на Трубачеева",
	
	"content": "Маленькая яма на дороге",
	
	"location": "Рядом с ТЦ \"Магнат\"",
	
	"latitude": 51.87860549,
	
	"longitude": 107.3343297,
	
	"img_link": "img_3.jpg",
	
	"author": {
	
	"team_name": "Организатор трека Vue.js",
	
	"description": "Тут моя краткая биография2",
	
	"track_id": 2,
	
	"track_name": "Фронтенд разработка на Vue.js ",
	
	"track_description": "Участники трека будут способны разработать фронтэнд приложение, которое будет работать с API. ",
	
	"user_name": "Хабитуев Баир Викторович",
	
	"updated_at": "2024-03-08T03:07:07.000000Z"
	
	},
	
	"report_type": {
	
	"id": 1,
	
	"created_at": "2024-03-21T12:29:55.000000Z",

	"updated_at": "2024-03-21T12:29:55.000000Z", "title": "Ямы", "description": "Ямы города Улан-Удэ", "tag": "pit" } } **Denied** Если сущность не существует (передан неверный идентификатор) Status: 404

Комментарии к заявке

GET (comments/{id}) - комментарии к заявке. Система принимает на вход идентификатор заявки и возвращает массив комментариев к заявке. При этом будут отправлены только публичные комментарии.

Если к запросу прикрепить Bearer токен, то система вернёт все комментарии.

Пример ссылки (для заявки с id=3):

https://webcomp.bsu.ru/api/finals24/reports/1

Request	**Response**

Content-type: application/json
 	
Parameters:
 	
в качестве параметра в строке передается id запроса.
 	
Опционально можно приложить Bearer токен
	Success:
	
	В базе данных существует запрос
	
	Status: 200/OK
	
	Content-type: application/json
	
	Body: "data": [
	
	{
	
	"id": 1,
	
	"created_at": "2024-03-23T08:29:15.000000Z",
	
	"updated_at": "2024-03-23T08:29:15.000000Z",
	
	"user_id": 1,
	
	"content": "Это очень большая проблема",
	
	"status": 1,
	
	"report_id": 1,
	
	"author": {
	
	"team_name": "Организатор трека Vue.js",
	
	"description": "Тут моя краткая биография",
	
	"track_id": 2,
	
	"track_name": "Фронтенд разработка на Vue.js ",
	
	"track_description": "..",
	
	"user_name": "Хабитуев Баир Викторович",
	
	"updated_at": "2024-03-08T03:07:07.000000Z"
	
	}

	}, ...] **Denied** если сущность не существует (передан неверный идентификатор) Status: 404

Типы заявок

GET (report/types/all) - типы заявок.

Request	**Response**

Content-type: application/json	**Success:**
	В базе данных существует запрос
	Status: 200/OK
	Content-type: application/json
	Body: "data": [
	{
	"id": 1,
	"title": "Ямы",
	"description": "Ямы города Улан-Удэ",
	"tag": "pit"
	},
	{
	"id": 2,
	"title": "Мусор",
	"description": "Несанкционированные …",
	"tag": "trash"
	}
	]
	Denied
	ошибка системы
	Status: 404

Авторизация

POST (login) - отправка данных авторизации. Система принимает на вход JSON массив с логином и паролем. Если логин и пароль указаны верно пользователь получает строку - Bearer токен.

Request	**Response**
Content-type: application/json Parameters: Передается объект: {“Login”:”cow@bsu.ru”,”Password”:”1”}	**Success:** *Пользователь есть в базе* Status: 200/OK Content-type: application/json Body: *Bearer-token* **Denied** Такого пользователя в базе нет Status: 404

Авторизованные запросы

Для выполнения авторизованных запросов в систему необходимо передать действующий Bearer токен, полученный после ввода логина и пароля в форме логина (в случае если логин и пароль были верно указаны). Используйте логины и пароли с отборочного тура.

UPD (17.05): для получения логина и пароля вам нужно **зарегистрировать команду.** Подробности в официальном телеграм канале конкурса.

Заявки

Добавление заявки

PUT (reports/add) - добавить заявку. Система принимает на вход JSON с данными заявки.

Request	**Response**

Content-type: application/json
Authorization: Bearer <токен>
Parameters:
Передается объект:
{
"title":"Текст заявки",
"report_type_id":1,
"content":"Содержание заявки",
"location":"На Трубачеева",
"latitude":60.30304,
"longitude":50.34045,
"img_link":"img_link"
}
Примечание:
Поля не должны быть пустыми.
report_type_id - обязательно идентификатор типов заявок
img_link по умолчанию пишем “stop.png”
	Success: *Заявка успешно добавлена*
	Status: 201/Created
	Система возвращает добавленную заявку
	Content-type: application/json
	"data": {
	"title":"Текст заявки",
	"report_type_id":1,
	"content":"Содержание заявки",
	"location":"На Трубачеева",
	"latitude":60.30304,
	"longitude":50.34045,
	"img_link":"img_link",
	"updated_at": "2024-03-24T01:56:34.000000Z",
	"created_at": "2024-03-24T01:56:34.000000Z",
	"id": 12
	}
	Body:
	errors:
	Status:

	302 - Неверно указан один из параметров

Редактирование заявки

PUT (reports/edit) - редактирование заявки. Пользователь может редактировать только свои заявки. Система принимает на вход JSON массив.

Request	**Response**

Content-type: application/json
 	
Authorization: Bearer <токен>
 	
Parameters:
 	
Передается объект:
 	
{
 	
"report_id":<id заявки>,
 	
"title":"Текст заявки",
 	
"report_type_id":1,
 	
"content":"Содержание заявки",
 	
"location":"На Трубачеева",
 	
"latitude":60.30304,
 	
"longitude":50.34045,
 	
}
 	
Примечание:
 	
content не должен быть пустым
 	
status может принимать значения 1 - публичный, 2 - только для авторизованных.
	Success: *комментарий отредактирован*
	
	Status: 200/OK
	
	Content-type: application/json
	
	Body:
	
	errors:
	
	Status:
	
	404 - Несуществующий id заявки (report_id)
	
	403 - Пользователь не является создателем заявки
	
	302 - Неверно указан один из параметров

Удаление заявки

GET (reports/delete/{id}) - удаление заявки. Пользователь может удалять только свои заявки. При удалении заявки автоматически удаляются все привязанные комментарии. Система принимает на вход id заявкив адресной строке.

Request	**Response**
Content-type: application/json Authorization: Bearer <токен> Parameters: В адресной строке передаётся id заявки	Success: *заявка удалёна* **Status:** 200/OK Content-type: application/json Body: *errors:* **Status:** 404 - Несуществующий id заявки 403 - Пользователь не является создателем заявки

GET (reports/get/all) - все заявки пользователя

Request	**Response**
Content-type: application/json Authorization: Bearer <токен>	Success: *комментарий удалён* **Status:** 200/OK Content-type: application/json Body: *data:[Массив заявок пользователя]* **Status:** 404 - неполадки в системе

Комментарии

Добавление комментария

PUT (comments/add) - добавить комментарий. Система принимает на вход JSON с данными комментария.

Request	**Response**

Content-type: application/json
 	
Authorization: Bearer <токен>
 	
Parameters:
 	
Передается объект:
 	
{
 	
"content":"Текст комментария",
 	
"status":2,
 	
"report_id":1
 	
}
 	
Примечание:
 	
content не должен быть пустым
 	
status может принимать значения 1 - публичный, 2 - только для авторизованных.
	Success: *Комментарий успешно добавлен*
	
	Status: 201/Created
	
	Система возвращает добавленный комментарий
	
	Content-type: application/json
	
	"data": {
	
	"content": "...",
	
	"status": 1,
	
	"report_id": 1,
	
	"user_id": 1,
	
	"updated_at": "2024-03-24T01:56:34.000000Z",
	
	"created_at": "2024-03-24T01:56:34.000000Z",
	
	"id": 12
	
	}
	
	Body:
	
	errors:
	
	Status:
	
	404 - Несуществующий id заявки (report_id)
	
	302 - Неверно указан один из параметров (content, status)

Редактирование комментария

PUT (comments/edit) - редактирование комментария. Пользователь может редактировать только свои комментарии. Система принимает на вход JSON массив.

Request	**Response**

Content-type: application/json
 	
Authorization: Bearer <токен>
 	
Parameters:
 	
Передается объект:
 	
{
 	
"comment_id":<id комментария>,
 	
"content":"Текст комментария",
 	
"status":2
 	
}
 	
Примечание:
 	
content не должен быть пустым
 	
status может принимать значения 1 - публичный, 2 - только для авторизованных.
	Success: *комментарий отредактирован*
	
	Status: 200/OK
	
	Content-type: application/json
	
	Body:
	
	errors:
	
	Status:
	
	404 - Несуществующий id комментария (comment_id)
	
	403 - Пользователь не является создателем комментария
	
	302 - Неверно указан один из параметров (content, status)

Удаление комментария

GET (comments/delete/{id}) - удаление комментария. Пользователь может удалять только свои комментарии. Система принимает на вход id комментария в адресной строке.

Request	**Response**
Content-type: application/json Authorization: Bearer <токен> Parameters: В адресной строке передаётся id комментария	Success: *комментарий удалён* **Status:** 200/OK Content-type: application/json Body: *errors:* **Status:** 404 - Несуществующий id комментария (comment_id) 403 - Пользователь не является создателем комментария

Комментарии пользователя

GET (comments/get/all) - все комментарии пользователя

Request	**Response**
Content-type: application/json Authorization: Bearer <токен>	Success: *комментарий удалён* **Status:** 200/OK Content-type: application/json Body: *data:[Массив комментариев пользователя]* **Status:** 404 - неполадки в системе