Описание системы классификации элементов BIM-моделей - Релиз 1.0

Система классификации элементов БИМ-моделей (далее Система) предназначена для классификации элементов BIM-моделей по заданной структуре каталогов.

Система работает в двух режимах: инференс и обучение.

В режиме **инференса** система получает на вход:

- BIM-модель
- структуру каталога, по которому нужно классифицировать ее элементы

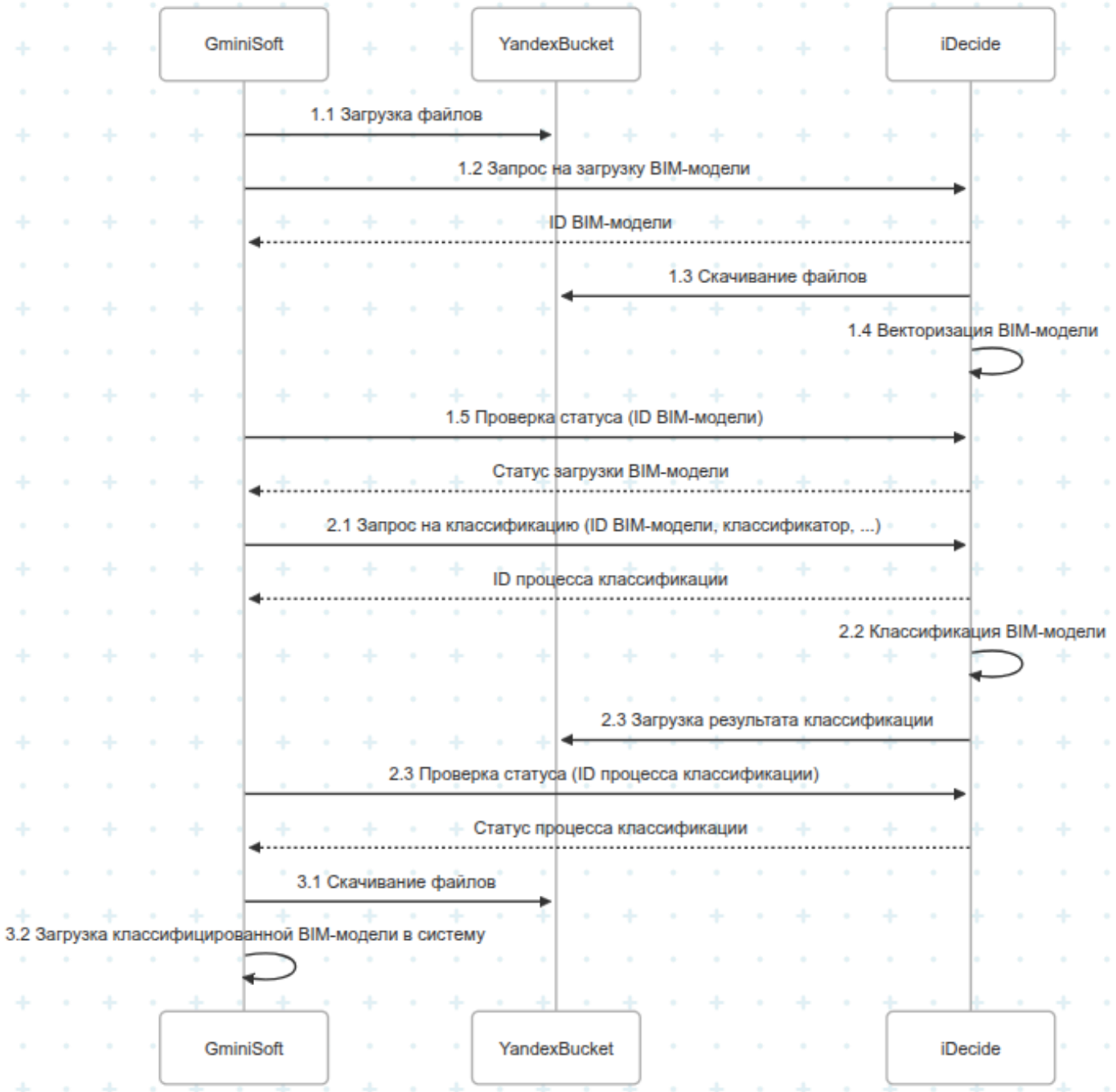
и на выходе возвращает список элементов BIM-модели с проставленными метками классов.

В режиме **обучения** система получает на вход обучающее множество в виде списка элементов BIM-модели с метками классов, производит дообучение и возвращает id обученной ИИ модели для дальнейшего использования в режиме инференса.

Система реализована в виде веб-приложения, развертываемого с помощью docker-compose и интерфейсом REST API. Передача файлов BIM-модели и результатов классификации осуществляется через Яндекс-бакет.

Схема взаимодействия с пользователем

Пользователем выступает приложение на стороне заказчика (обозначено на схеме как GminiSoft). Она обращается к Системе (обозначено на схеме как iDecide) с помощью REST API.



Процесс взаимодействия выглядит следующим образом:

- Пользователь загружает BIM-модель в Яндекс бакет (1.1).
- Пользователь отправляет запрос на обработку БИМ модели в Систему (1.2).

- Система начинает процесс загрузки из Яндекс бакета и векторизацию BIM-модели (1.3, 1.4) и возвращает Пользователю идентификатор процесса загрузки, по которому он может запросить статус обработки (1.5).
- Когда обработка завершена (статус загрузки Success), пользователь может отправить в систему запрос на классификацию загруженной BIM-модели по каталогу. (2.1) Структура каталога передается в теле запроса.
- Система запускает процесс классификации (2.2) и возвращает Пользователю идентификатор процесса классификации, по которому он может запросить статус классификации (2.3).
- По окончании классификации Система выкладывает результаты классификации в Яндекс бакет, и Пользователь может забрать их (3.1) и использовать в своих целях (3.2).

Формат входных и выходных данных

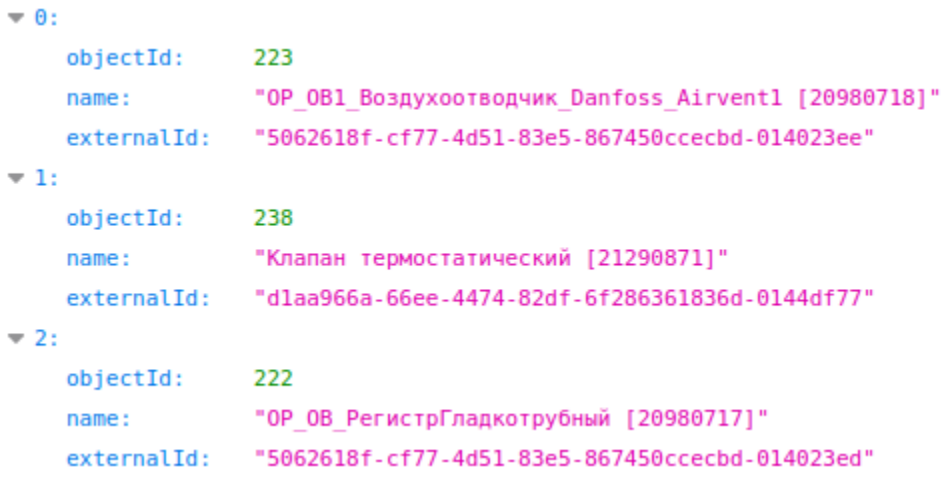
BIM-модель

Передается через Яндекс Бакет.

Задается в виде трех json файлов, описывающих элементы BIM-модели, их свойства, и связи между ними:

- nodes.json
- properties.json
- edges.json

Пример **nodes.json** (фрагмент)**:**



Пример **properties.json** (фрагмент):

Пример **nodes.json** (фрагмент):

Структура каталога

Передается в виде json в теле запроса REST API.

Описывает иерархическую структуру папок, по которым нужно разложить элементы BIM-модели. Пример структуры каталога:

```
[
    {"id": 1, "name": "root", "parent": None},
    {"id": 2, "name": "клапаны", "parent": 1},
    {"id": 3, "name": "клапаны гидростатические", "parent": 2},
    {"id": 4, "name": "клапаны гидравлические", "parent": 2},
    {"id": 5, "name": "воздухоотводчики", "parent": 1}
]
```

Результаты классификации

Передаются через Яндекс бакет.

Представляют собой файл csv со следующими полями:

- element_id: str - id элемента BIM-модели
- folder_id: str - id папки из каталога
- element_desc: str - текстовое описание элемента
- folder_desc: str - текстовое описание папки
- confidence:float - уверенность классификатора от 0 до 1

Пример:

element_id	folder_id	element_description	folder_description	confidence
123	5	{'name': 'OP_OB1_Воздухоотводчик_Danfoss_	root -> воздухоотводчики	0.9
432	5	{'name': 'OP_OB1_Воздухоотводчик_Danfoss_	root -> воздухоотводчики	0.9
543	6	{'name': '532_santehprom_radiator_rbs-500-90	root -> клапаны -> клапаны гидростатиче	0.85
563	12	{'name': '534_santehprom_radiator_rbs-500-90	root -> клапаны -> клапаны гидравличес	0.87

Данные для дообучения

Передаются через Яндекс бакет.

Данные для дообучения аналогичны результатам классификации, но вместо confidence содержат два других столбца:

- label: bool. - 1 если классификация верна и 0 если не верна. Проставляется человеком
- comment: str - комментарий специалиста (опционально)

element_id	folder_id	element_description	folder_description	label	comment
123	5	{'name': 'OP_OB1_Воздухоотводчик_Danfoss_	root -> воздухоотводчики	1	
432	5	{'name': 'OP_OB1_Воздухоотводчик_Danfoss_	root -> воздухоотводчики	1	
543	6	{'name': '532_santehprom_radiator_rbs-500-90	root -> клапаны -> клапаны гидростатиче	1	
563	12	{'name': '534_santehprom_radiator_rbs-500-90	root -> клапаны -> клапаны гидравличес	0	ошибочная классификация, потому что...

Описание API

Детальная спецификация API со схемами данных в формате OpenAPI прилагается отдельно.

Перечень эндпоинтов:

1. Управление BIM-моделями
    a. **Загрузка BIM-модели.** Параметры: файл BIM-модели, версия ИИ-модели. Производит загрузку и векторизацию BIM-модели. Возвращает идентификатор BIM-модели.
    b. **Проверка статуса загрузки BIM-модели.** Параметры: идентификатор BIM-модели. Возвращает статус обработки (готово \ текущий прогресс \ ошибка).
    c. **Получение списка BIM-моделей.** Возвращает список идентификаторов BIM-моделей и их размер.
    d. **Удаление BIM-модели.** Параметры: идентификатор BIM-модели. Удаляет BIM-модель.
2. Классификация BIM-модели
    a. **Запуск классификации BIM-модели.** Параметры: идентификатор BIM-модели, структура каталога, версия ИИ-модели. Запускает процесс классификации и возвращает ID этого процесса.
    b. **Проверка статуса классификации BIM-модели.** Параметры: идентификатор процесса классификации. Возвращает статус (готово \ текущий прогресс \ ошибка).
    c. **Остановка запущенного процесса классификации BIM-модели.** Параметры: идентификатор процесса классификации.
    d. **Получение результатов классификации BIM-модели.** Параметры: идентификатор процесса классификации. Возвращает файл с результатами классификации.
    e. **Получение списка результатов классификации BIM-моделей.** Возвращает список всех ранее запущенных процессов классификации BIM-моделей с атрибутами: ID, статус, дата/время запуска и завершения.
3. Дообучение BIM-модели
    a. **Запуск дообучения BIM-модели.** Параметры: идентификатор BIM-модели, структура каталога, данные для дообучения, версия ИИ-модели. Возвращает: идентификатор новой дообученной ИИ-модели.
    b. **Проверка статуса дообучения BIM-модели.** Параметры: идентификатор дообученной ИИ-модели. Возвращает статус (готово \ текущий прогресс \ ошибка). Если модель успешно дообучена, для ее использования нужно передавать ее новый идентификатор в методе 5.
4. **Получение списка ИИ-моделей.**

Логика работы ИИ

Загрузка и предобработка BIM-модели

Состоит из следующих шагов:

1. Составление текстовых описаний элементов BIM-модели
2. Векторизация. Построение эмбеддингов элементов BIM-модели с помощью LLM
3. Кластеризация элементов BIM-модели
4. Сохранение в векторное хранилище

Классификация элементов BIM-модели по каталогу

Состоит из следующих шагов:

1. Предобработка и векторизация структуры каталога
2. Определение оптимальных пороговых значений близости
3. Поиск кандидатов на основе векторной близости между эмбеддингом элемента БИМ-модели и эмбеддингом директории каталога
4. Двухшаговая валидация кандидатов с помощью LLM:
    a. Предварительная валидация без ризонинга
    b. Для кандидатов прошедших предварительную валидацию - финальная валидация с ризонингом
5. Разрешение неоднозначностей в случае многоклассовой классификации
6. Распространение результатов на все элементы кластера
7. Сохранение результатов классификации

Дообучение ИИ модели

В релизе 1 реализуется на основе векторного поиска + RAG:

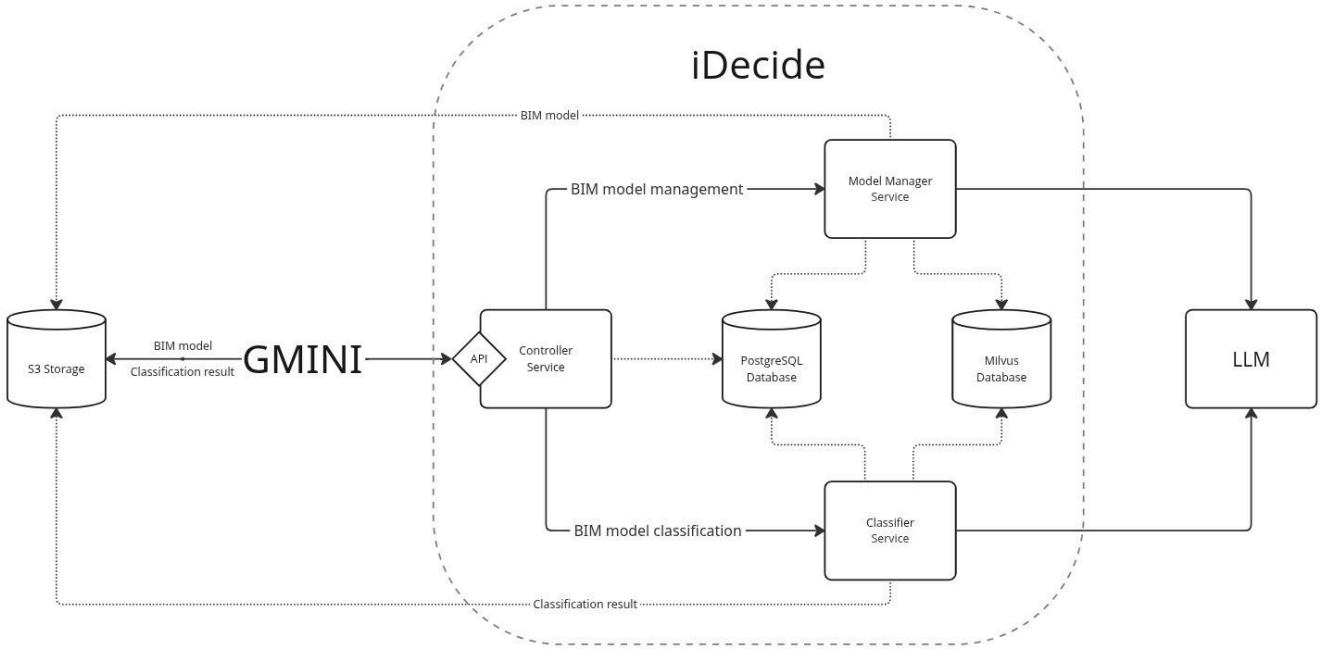
Логика подхода:

При попытке классифицировать элемент BIM-модели А в папку Б осуществляется поиск, не было ли похожего случая в обучающих данных. Если найден очень похожий случай (близость выше HIGH_THRESHOLD), то классификация производится аналогично этому случаю. Если найден не очень похожий случай (близость выше LOW_THRESHOLD но ниже HIGH_THRESHOLD), то применяется RAG.

RAG работает следующим образом. Допустим, найден похожий случай классификации элемента А1 в похожую папку Б1, и он был отмечен человеком как ошибочный. Эта информация добавляется в промпт, чтобы ЛЛМ приняла ее во внимание.

В дальнейших релизах по мере накопления обучающих данных планируется реализация на основе файнтюна весов LLM.

Физическая архитектура



Hardware компоненты

Для функционирования системы необходимы следующие аппаратные компоненты:

- Сервер для приложения и баз данных
- Сервер файлового S3 хранилища

Software компоненты

В первой версии система состоит из следующих компонентов:

- S3 Storage - файловый сервис для хранения BIM моделей и результатов их классификации
- Controller Service - сервис управления процессами системы и интеграции с Gmini системой
- Model Management Service - сервис управления жизненным циклом BIM моделей
- Classifier Service - сервис классификации BIM моделей
- PostgreSQL DB - хранилище метаданных системы
- Milvus DB - векторное хранилище данных системы (изменение: вместо Milvus решено использовать Clickhouse DB)

- LLM - внешняя LLM, участвующая в процессах системы (в первой версии используется OpenAI)

Системные требования

Требования к окружению:

- Docker v27+
- Docker compose v2.37+

Минимальные требования к оборудованию:

- 16Gb оперативной памяти
- 50Gb свободного места на диске
- Широкополосный доступ в интернет

Прочие требования:

- Наличие доступа к OpenAI API с привязанным биллингом

Ограничения Релиза 1.0

- Однопользовательский режим работы с API системы
- Открытое API системы без какой-либо аутентификации
- Одновременно можно загружать только одну BIM-модель
- Одновременно можно классифицировать только одну BIM-модель
- Отказоустойчивость системы обеспечивается сервисами докера
- Обновление системы происходит аналогично запуску системы
- При обновлении системы временем простоя является время между запуском обновления и готовности всех сервисов по окончанию обновления