Реализация системы бэкапа игровых сохранений

Основное приложение будет разработано с помощью Net Maui или WPF.

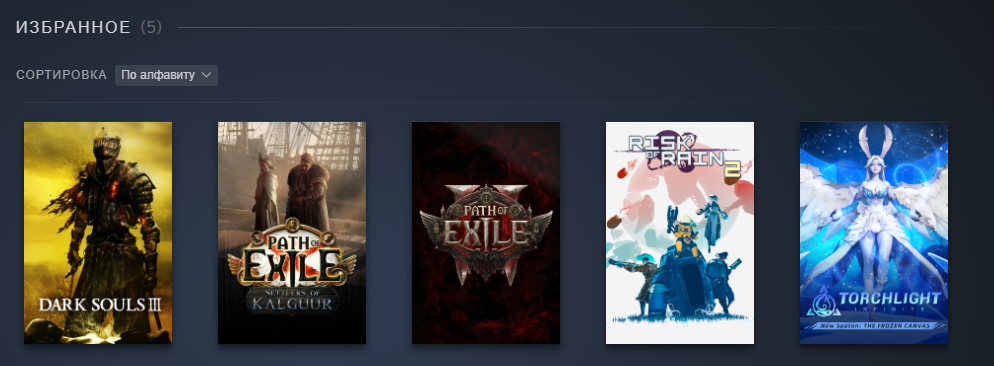
На Laravel будет сделано API. В БД будут храниться популярные игры и пути сохранения файлов и картинка.  
Также будет сайт на Laravel, который будет красиво отображать сохранённые данные.  
Будет как коллекции игр в стиме. 

Рисунок 1

При нажатии на неё будет полоска сверху дата и время

Чуть ниже сам файл/папка или архив

Как на рисунке 1 справа от избранного будет публичность (да или нет).

Бэкап после каждого изменения

**ОТВЕТ ОТ CHATGPT**

Реализация системы бэкапа игровых сохранений — интересная задача, но действительно требует продуманной архитектуры. Вот пошаговый подход, как это можно организовать:

**1. Сбор данных о местоположении сохранений**

Чтобы приложение могло автоматизировать процесс поиска сохранений, нужно собрать данные о:

* **Популярных играх**: Список игр, их версий и платформ.
* **Пути к сохранениям**: Обычно это фиксированные директории (например, Documents, AppData или корневая папка игры).
* **Типы файлов**: Какие файлы относятся к сохранениям (например, .sav, .bin, .profile).

**Реализация**:

* Сначала создаётся небольшая база данных популярных игр и их путей.
* Эти данные можно взять из открытых источников (например, форумов или сайтов со справочной информацией).
* Пользователи могут добавлять пути вручную, если игра не найдена в базе.

Пример структуры таблицы в БД:

plaintext

Копировать код

Игры

- ID (int)

- Название игры (string)

- Версия (string, optional)

- Платформа (PC, PS5, Xbox)

- Путь к сохранениям (string)

- Типы файлов (string, например, "\*.sav;\*.bin")

**2. Поиск сохранений на устройстве**

Приложение должно сканировать указанные директории.

* Например, Windows игры часто хранят данные в:
  + %USERPROFILE%\Documents
  + %USERPROFILE%\AppData\Local
  + %USERPROFILE%\AppData\Roaming
* Для консольных игр путь будет менее гибким, но возможен импорт сохранений вручную.

**Реализация**:

* Сканер, который ищет папки и файлы по известным путям из базы данных.
* Пользователь вручную подтверждает найденные сохранения.

**3. Резервное копирование в облако**

После обнаружения сохранений их можно загрузить в облачное хранилище.

* Поддержка шифрования файлов перед загрузкой для безопасности.
* Ведение истории версий (чтобы можно было восстановить прошлую версию).
* Возможность указать, какие файлы загружать.

**Реализация**:

* Использовать Laravel для создания API и управления облачным хранилищем.
* Хранить сохранения в облаке (например, на AWS S3, Google Cloud или собственном сервере).

Пример API:

* POST /upload — загрузить файл.
* GET /files — получить список файлов.
* DELETE /file/{id} — удалить файл.

**4. Восстановление сохранений**

Пользователь может загрузить сохранения обратно на своё устройство.

* Автоматическое восстановление в нужную директорию (если приложение имеет права).
* Или предоставление архива для ручного восстановления.

**Реализация**:

* Опция "Восстановить" в интерфейсе.
* Скачивание файла через API, затем копирование его в нужный путь.

**5. Добавление игр пользователем**

Поскольку база данных всех игр невозможна, можно позволить пользователям добавлять свои игры:

* Указать путь к папке сохранений.
* Задать расширения файлов для копирования.

**Реализация**:

* Форма для добавления игры в личную базу.
* Возможность делиться данными о пути с другими пользователями (модерация данных).

**6. Расширенный функционал (по желанию)**

* **Интеграция с игровыми платформами**: Например, Steam API для отображения списка установленных игр.
* **Графическое представление истории бэкапов**: В виде графиков, показывающих, как часто обновляются сохранения.
* **Уведомления**: Напоминания о необходимости сделать новый бэкап.
* **Шифрование**: AES-256 для повышения безопасности.

**Инструменты для реализации**

**Бэкенд (Laravel)**

* API для управления файлами (загрузка, скачивание, удаление).
* Взаимодействие с базой данных (хранение метаинформации об играх и файлах).

**Клиентская часть (MAUI)**

* Сканирование локальных директорий.
* Отправка и получение данных через API.
* Локальная база данных SQLite для хранения путей и настроек.

**База данных**

* SQLite для локального клиента.
* MySQL/PostgreSQL на сервере для хранения глобальных данных.

**Уникальность**

* Нет аналогов, которые фокусируются только на сохранениях.
* Возможность добавить поддержку пользовательских игр делает сервис универсальным.
* Можно выделиться на фоне крупных облачных сервисов благодаря специализированному функционалу.

Если заинтересовала такая тема, можем более детально расписать архитектуру!

4o

**Вы сказали:**

я поискал в интеренет и есть уже GameSave Manager. Что у меня будет такого нового, чего не будет у него?

**ChatGPT сказал:**

ChatGPT

GameSave Manager действительно уже существует, и он автоматизирует резервное копирование сохранений. Однако у вас есть возможность добавить функции и отличительные особенности, которых у этого инструмента нет. Вот несколько идей, как сделать ваш проект уникальным и функциональным:

**1. Поддержка кросс-платформенных сохранений**

GameSave Manager фокусируется на ПК. Вы можете расширить функционал, добавив:

* **Импорт сохранений с консольных платформ**: PlayStation, Xbox, Nintendo Switch.
* **Синхронизацию между платформами**: Например, перенос сохранений из консоли на ПК (при условии, что игра поддерживает это).
* **Интеграция с облачными сервисами** (Google Drive, OneDrive, Dropbox), чтобы пользователь мог выбирать хранилище.

**2. Полноценная история и версионирование сохранений**

Добавьте функционал, позволяющий:

* Автоматически сохранять версии файлов при каждом обновлении.
* Позволять пользователю откатываться к предыдущей версии сохранения.
* Визуализировать историю бэкапов в виде временной шкалы.

**3. Улучшенная интеграция с платформами (Steam, Epic, GOG)**

* Интеграция через API Steam/Epic/GOG для автоматического определения установленных игр.
* Получение списка игр, включая те, что поддерживают облачные сохранения.
* Возможность отображения игровой статистики (например, время в игре) на одной платформе.

**4. Расширенная аналитика сохранений**

Создайте инструмент, который анализирует сами файлы сохранений, например:

* Отображает ключевую информацию из сохранения (например, текущий прогресс, уровень, инвентарь).
* Распознаёт проблемы с сохранениями (повреждённые файлы).

**5. Совместный доступ к сохранениям**

Позвольте пользователям делиться сохранениями с друзьями через ваш сервис.

* Функция "Поделиться сохранением" с генерированием ссылки.
* Возможность загрузки чужого сохранения для конкретной игры.

**6. Уведомления и расписание бэкапов**

* Настройка регулярных бэкапов по расписанию.
* Уведомления о необходимости сделать новый бэкап после значительных изменений в сохранении.
* Уведомления о новых поддерживаемых играх.

**7. Резервное копирование модов**

GameSave Manager не фокусируется на модах. Вы можете добавить:

* Автоматическое копирование папок с модами вместе с сохранениями.
* Поддержка популярных мод-хранилищ (например, NexusMods, Steam Workshop).

**8. Пользовательская база данных игр**

Дайте пользователям возможность расширять базу данных, добавляя свои пути к сохранениям.

* Платформа для обмена конфигурациями путей к играм.
* Возможность подтверждения и оценки добавленных данных сообществом.

**9. Расширенный функционал для мобильных платформ**

Поддержка игр на мобильных устройствах (Android, iOS):

* Резервное копирование сохранений из мобильных игр.
* Перенос сохранений между устройствами.

**10. Гибкая настройка шифрования и безопасности**

GameSave Manager не делает акцента на безопасности. Вы можете добавить:

* Локальное и облачное шифрование данных перед загрузкой.
* Защиту доступа к сохранениям паролем или биометрией.

**Чем ваш проект будет отличаться?**

1. **Кросс-платформенность**: Поддержка ПК, консолей, мобильных устройств.
2. **Интеграция с облачными платформами**: Поддержка синхронизации с Google Drive, OneDrive, Dropbox.
3. **Расширенные функции аналитики и модов**: Возможность работы с модами и анализа самих сохранений.
4. **Социальные функции**: Совместное использование сохранений, комьюнити-ориентированная база данных.
5. **Гибкость и расширяемость**: Пользователь может добавить свои игры и пути, чтобы покрыть весь спектр потребностей.

Если что-то из этих идей заинтересовало, могу помочь спланировать реализацию или предложить более детальный функционал.