

TVPlay: audyt, modernizacja i automatyzacja emisji

TVPlay (Open Playout Automation) to otwartoźródłowy system automatyzacji emisji telewizyjnej, stworzony i używany produkcyjnie w ośrodkach regionalnych TVP. Steruje serwerem wideo CasparCG: odtwarzaniem materiałów, grafiką antenową i ramówką. Rozwijam go na własnym stanowisku emisyjnym, pracując z agentami AI: od pełnego audytu kodu, przez automatyczne układanie ramówki, po samoobsługową rotację materiałów na dyskach.

Autor	Marcin Jabłoński, projekt własny
Okres	2026: audyt i naprawy od 18.08, rozbudowa ramówki we wrześniu, automatyzacja folderu EMISJA od 7.10
Technologie	C#, .NET Framework 4.8, WPF, CasparCG Server 2.3.2 LTS, SQLite, REST API wtyczki; narzędzia pomocnicze w Pythonie
Praca z AI	Claude Code: równoległe przebiegi analizy, wdrażanie zmian, wieloagentowa recenzja kodu; weryfikacja kompilacją, testami na prawdziwej konfiguracji i logami
Projekt bazowy	Open Playout Automation (jaskie/PlayoutAutomation), licencja GPL v2

53 600 linii kodu C# w 23 projektach przeanalizowanych w audycie	171 usterek znalezionych i naprawionych, w tym 2 krytyczne	35 → 0 odstępów ponad 10 minut w tygodniu ułożonym automatycznie	250 nowych odcinków (201,6 GB) dołączonych z kontrolą sum SHA-256
--	--	--	---

1. Audyt kodu pod kątem pracy 24/7 bez operatora

Cel: sprawdzić, czy system wytrzyma całodobową emisję bez nadzoru. Analizę wykonało 7 równoległych przebiegów agentów AI; każdy czytał w całości pliki jednego podsystemu. Każde znalezisko ma wskazane miejsce w kodzie (plik i linia) oraz scenariusz awarii. 11 najpoważniejszych, w tym oba krytyczne, sprawdziłem dodatkowo bezpośrednio w kodzie źródłowym: wszystkie się potwierdziły.

Waga	Liczba	Podsystem	Usterki
Krytyczna: przerwa w emisji lub utrata danych	2	Silnik emisyjny	24
		Media i pobieranie materiałów	30
Wysoka: realna awaria w pracy 24/7	27	Wtyczki (krosownice, grafika, ramówka)	26
Średnia: błędne działanie, wycieki zasobów	73	Klient operatora	32
		Wspólny interfejs i konfiguracja	16
Niska: drobne błędy i niespójności	69	Baza danych i warstwa wspólna	22
		Komunikacja z serwerem wideo	21
Razem	171		

Przykłady

- Krytyczna:** zapytanie „czy materiał jest zaplanowany?” porównywało w bazie SQLite liczbę z tekstem daty, więc zawsze odpowiadało „nie”. Operator mógł skasować klip zaplanowany na wieczór, a ramówka dochodziła do pustego miejsca.

- **Krytyczna:** jedyny wątek obsługi komunikatów klienta mógł się zakleszczyć, czekając na odpowiedź, której sam miał udzielić.
- **Wysoka:** REST API wtyczki ramówki nasłuchiwało na wszystkich interfejsach sieciowych bez uwierzytelniania. Teraz domyślnie działa tylko lokalnie, dostęp z sieci wymaga klucza, a zapis plików jest ograniczony do jednego folderu.
- **Wysoka:** tekst wyszukiwania w archiwum trafiał wprost do zapytania SQL (ryzyko wstrzyknięcia kodu i awarii na cudzysłowie). Teraz zapytanie jest parametryzowane.

Weryfikacja. Wszystkie 171 usterek naprawiono; kompilacja całego rozwiązania kończy się bez błędów C#, a liczba ostrzeżeń spadła z 63 do 56. Następnie gałąź z naprawami przeszła wieloagentową recenzję w chmurze (136 plików). Trzy kolejne znaleziska, m.in. rekurencję grożącą przepełnieniem stosu przy długich playlistach i nieatomowy zapis konfiguracji, sprawdziłem w kodzie i naprawiłem. Zmiany, które celowo zmieniają zachowanie programu (np. API domyślnie tylko lokalnie, 60 sekund na odpowiedź serwera), opisałem osobno, żeby aktualizacja nie zaskoczyła operatora.

Liczba usterek nie przekreśla programu, który działa produkcyjnie w telewizji. Większość dotyczy pracy bez operatora, w której błąd mógłby przerwać emisję niezauważony.

2. Automatyczna ramówka

- **Struktura przerw:** pięć pul materiałów (identy, zapowiedzi, reklamy, wypełniacze, bilbordy) i szablony przerw dobierane do długości luki, z limitem minut reklam na godzinę.
- **Rotacja reklam jak talia kart:** materiał wraca dopiero po rozdaniu całej talii, a stan zapisuje się atomowo po każdym wygenerowanym dniu. Wcześniej pula była tasowana od nowa w każdym bloku, więc te same reklamy często się powtarzały.
- **Układ tygodnia:** wymaganie brzmiało „bez odstępów między pozycjami dłuższych niż 5-10 minut i ze stałymi godzinami seriali nadawanych częściej niż 2 razy w tygodniu”. Algorytm (blok dnia, doklejanie pasm na końcach dni, wyżarzanie) dopracowałem z AI najpierw w symulatorze w Pythonie (cztery wersje), a potem przeniósłem do C#. Na prawdziwej konfiguracji 39 pasm: zamiast 35 odstępów ponad 10 minut (najdłuższy 162 min) zero takich odstępów, każde pasmo o jednej stałej godzinie, wynik w 1,4 s.
- **Pozostałe:** doba ramówkowa, powtórki, eksport programu (EPG, XMLTV), grafiki „za chwilę / następnie” i pasek wiadomości, który co 5 minut pobiera 10 najgorętszych tematów z mojego serwisu Newsy.

3. Automatyzacja folderu EMISJA

Problem. Na szybkim dysku emisyjnym leżały wszystkie odcinki (ok. 190 GB). Cel: na dysku tylko programy z dzisiejszej i jutrzejszej doby ramówkowej, archiwum na osobnym dysku 22 TB, przenoszenie w przerwie między dobami i możliwość podmiany ramówki w trakcie dnia.

Kluczowa obserwacja z analizy kodu. Usunięcie pliku przy działającym systemie kasuje jego rekord w bazie, a plik dodany ponownie dostaje nowy identyfikator. Zwykle przenoszenie plików zerwałoby więc powiązania w ramówkach wygenerowanych naprzód. Przetrwać może tylko rekord ze statusem „Wymagany”.

- **API rotacji we wtyczce:** stan, biblioteka, materiały zaplanowane, „parkowanie” (status „Wymagany”, który zachowuje rekord po usunięciu pliku i restarcie) oraz weryfikacja. Parkowanie jest odrzucane dla materiału na antenie, następnego i zaplanowanego w ciągu 3 godzin. Rdzeń systemu pozostał bez zmian.
- **Serwis rotacji** (Python, bez zewnętrznych bibliotek, startuje z systemem): na początku pętli kończącej dobę doładowuje z archiwum odcinki na dwie kolejne doby i odkłada niepotrzebne; co 15 sekund sprawdza, czy po podmianie ramówki czegoś nie brakuje na dziś i jutro.
- **Bezpieczeństwo danych:** kopie przez plik tymczasowy, sumę SHA-256 i atomową podmiannę nazwy; plik znika z dysku emisyjnego tylko wtedy, gdy ma sprawdzoną kopię w archiwum i zgodę wtyczki. Bez

odpowiedzi API nic nie jest usuwane.

- **Testy:** planowanie sprawdzone na migawce bazy z atrapą API i na działającym systemie (okno 6-8.10.2026: 63 pliki potrzebne, 364 pliki, 172,7 GB, do archiwum).
- **Konsolidacja biblioteki:** indeks ok. 79 tys. plików wideo ze wszystkich dysków, dopasowanie nazw serii, wykluczanie duplikatów po numerze odcinka, identyfikatorze nagrania i czasie trwania, wątpliwe pary sprawdzone na klatkach. Wynik: 250 nowych odcinków (201,6 GB) w 37 seriach. Dziennik CSV z sumami SHA-256 pozwala cofnąć każdą operację i posłużył do tego, gdy jedna seria miała wrócić na swoje miejsce.

Stan: rotacja działa codziennie od 7.10.2026. Pierwsza przeniosła do archiwum 350 plików; kolejne doładowują 18-25 odcinków i odkładają 16-25 (log z 8-11.10.2026).

4. Wdrożenie na nowym komputerze

Przeniesienie systemu (28.09.2026): 5052 pliki programu i 406 plików materiałów (177,5 GB) ze zgodną liczbą plików, bajtów i sumami SHA-256; instalacja CasparCG 2.3.2 LTS ze sprawdzeniem sumy kontrolnej; nowa baza utworzona własnym narzędziem; test ramówki z autostartem zakończony bez ostrzeżeń i błędów w logu.

5. Jak pracowałem z AI

- Wymagania formułowałem językiem operatora („bez odstępów ponad 10 minut”, „stałe godziny seriali”) i zamieniałem je w mierzalne testy na prawdziwej konfiguracji.
- Agenci AI analizowali kod i wdrażali zmiany; ja ustalałem zakres, sprawdzałem wyniki i zatwierdzałem wdrożenia, zawsze z kopią poprzedniej wersji.
- Przed zmianami w danych: kopia bazy i kontrola integralności; operacje na plikach z dziennikiem, który pozwala je odwrócić.
- Dokumentację prowadziłem na bieżąco: rejestr usterek (930 linii), rejestr napraw i rozbudowy (641 linii) oraz instrukcja obsługi w 22 rozdziałach.

Wersja publiczna: bez danych konfiguracyjnych i dostępowych, nazwy kanału oraz tytułów materiałów. Projekt niezależny, bez afiliacji z autorami systemu ani z TVP. Kontakt: kontakt@marcinjablonski.com.pl · marcinjablonski.com.pl/ai.html