MystDoku

MystDoku je interaktivní hra kombinující sudoku a příběhové prvky, kde hráč postupně odemyká vzpomínky. Tento dokument popisuje architekturu, herní logiku a správu dat, včetně administrace, bezpečnosti a generování herních mřížek.

Dokumentace projektu

# MystDoku – Dokumentace projektu

## Seznam kapitol

FAZE 1 – Přehled aplikací

FAZE 2 – Modely a jejich vztahy

FAZE 3 – Propojení URL → View → Template

FAZE 4 – Tok dat mezi částmi systému

FAZE 5 – Šablony a vykreslování

FAZE 6 – Herní logika a pravidla

FAZE 7 – Přístupová práva a zabezpečení

FAZE 8 – Validace a ladicí funkce

# Architektura projektu MystDoku

Verze: 1.0  
Datum: 2025-04-10  
Autor: Michal Pešta

# FAZE 1 – Přehled aplikací

Projekt MystDoku je rozdělen do více Django aplikací, z nichž každá plní specifickou roli v architektuře celého systému. Níže je přehled jednotlivých aplikací a jejich základní funkce.

## main

* Zajišťuje hlavní stránku, login, registraci a navigaci.
* Obsahuje globální context pro šablony (context\_processors.py).

## gameplay

* Starost o herní logiku: mřížky, místnosti, položky.
* Obsahuje modely Room, Item, později Game.
* Řeší start a průběh hry.

## score

* Sleduje statistiky hráčů a nejlepší časy.
* Obsahuje model PlayerScore, API a pomocné funkce.

# FAZE 2 – Modely a jejich vztahy

Tato část dokumentace popisuje datové modely jednotlivých aplikací a vztahy mezi nimi. Každý model je doplněn o seznam polí a jejich typ, případně napojení na jiné modely přes cizí klíče.

## Aplikace: gameplay

#### Room

Popis: Tematická místnost, do které patří různé položky.

* Pole:
* - name: CharField – název místnosti

#### Item

Popis: Herní položka (např. vidlička, hrnek), reprezentující číslo 1–9.

* Pole:
* - name: CharField – název položky
* - number: IntegerField – číslo položky (1–9)
* - group\_id: CharField – skupina významově stejných položek
* - room: ForeignKey → Room – vazba na místnost

#### Game

Popis: Instance rozehrané hry – obsahuje hráče, místnosti, stav a obtížnost.

* Pole:
* - id: UUIDField – unikátní ID hry
* - player: ForeignKey → User – hráč, kterému hra patří
* - created\_at: DateTimeField – datum vytvoření hry
* - completed: BooleanField – označuje, zda je hra dohraná
* - block\_rooms: JSONField – seznam místností (ID) pro jednotlivé bloky
* - block\_items: JSONField – mapování blok → číslo → item ID
* - difficulty: CharField – easy / medium / hard

#### Cell

Popis: Jedna buňka v sudoku mřížce – obsahuje správnou i vybranou položku.

* Pole:
* - game: ForeignKey → Game – hra, do které buňka patří
* - row: IntegerField – řádek (0–8)
* - column: IntegerField – sloupec (0–8)
* - correct\_item: ForeignKey → Item – správná položka
* - selected\_item: ForeignKey → Item (nullable) – hráčem vybraná položka
* - prefilled: BooleanField – zda je políčko předvyplněné

#### Intro

Popis: Sekvence úvodních textů, které se zobrazují při startu hry.

* Pole:
* - order: PositiveIntegerField – pořadí textu
* - text: TextField – obsah textu

#### Memory

Popis: Vzpomínky hlavního hrdiny, rozdělené podle obtížnosti a pořadí.

* Pole:
* - difficulty: CharField – easy / medium / hard
* - order: PositiveIntegerField – pořadí vzpomínky
* - text: TextField – hlavní text vzpomínky
* - transition: TextField – přechodový komentář (reakce hrdiny)

#### DifficultyTransition

Popis: Text zobrazený při přechodu mezi obtížnostmi (např. po získání 20 vzpomínek).

* Pole:
* - difficulty: CharField – označení fáze (easy / medium / hard / end)
* - text: TextField – zobrazený přechodový text

#### PlayerStoryProgress

Popis: Ukládá, které vzpomínky si hráč již odemkl podle obtížnosti.

* Pole:
* - player: OneToOneField → User – hráč
* - unlocked\_easy: JSONField – seznam ID odemčených vzpomínek (easy)
* - unlocked\_medium: JSONField – seznam ID odemčených vzpomínek (medium)
* - unlocked\_hard: JSONField – seznam ID odemčených vzpomínek (hard)

#### SequenceFrame

Popis: Obrázkový rámec v příběhové sekvenci (např. intro, memory...)

* Pole:
* - sequence: CharField – název sekvence
* - index: PositiveIntegerField – pořadí snímku
* - image: CharField – název obrázku (např. bg1.jpg)

# Aplikace: score

#### PlayerScore

Popis: Statistiky hráče – počet her, časy, pokrok v příběhu.

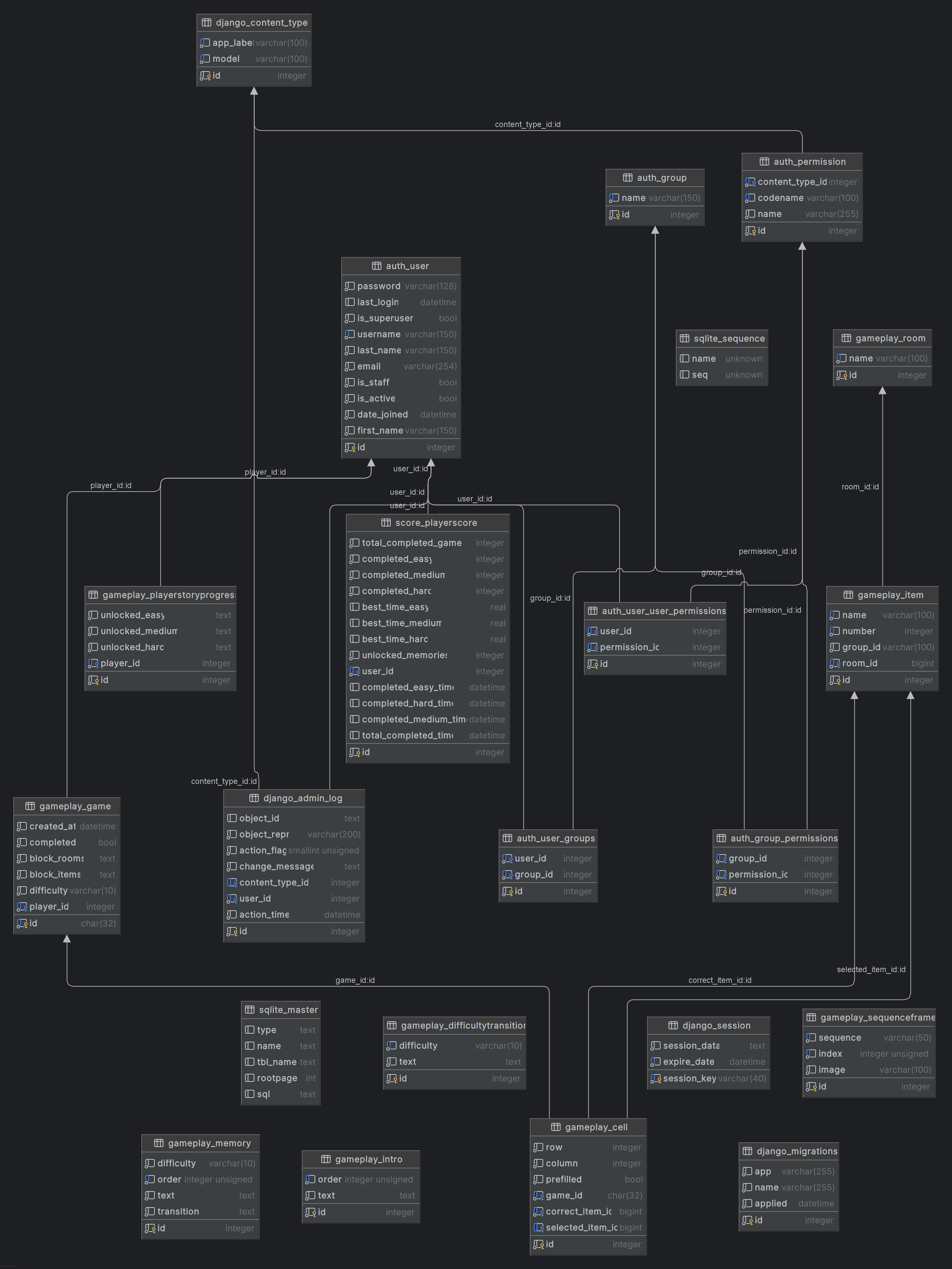
* Pole:
* - user: OneToOneField → User – propojení s hráčem
* - total\_completed\_games: IntegerField – celkový počet dokončených her
* - total\_completed\_time: DateTimeField – čas první dokončené hry
* - completed\_easy: IntegerField – počet dokončení na easy
* - completed\_easy\_time: DateTimeField – čas posledního easy dokončení
* - completed\_medium: IntegerField – počet dokončení na medium
* - completed\_medium\_time: DateTimeField – čas posledního medium dokončení
* - completed\_hard: IntegerField – počet dokončení na hard
* - completed\_hard\_time: DateTimeField – čas posledního hard dokončení
* - best\_time\_easy: FloatField – nejlepší čas v sekundách (easy)
* - best\_time\_medium: FloatField – nejlepší čas v sekundách (medium)
* - best\_time\_hard: FloatField – nejlepší čas v sekundách (hard)
* - unlocked\_memories: IntegerField – celkový počet odemčených vzpomínek (1–60)

## Poznámka k aplikaci main

Aplikace `main` sama o sobě neobsahuje žádné vlastní modely, ale využívá vestavěné modely frameworku Django, především pro správu uživatelů a autentifikaci. Patří sem zejména:

* - `User` model z `django.contrib.auth.models`
* - Session a autentifikační systém Django
* - Funkcionalita pro registraci, přihlášení, odhlášení

Tato funkcionalita je důležitá pro správné fungování celé aplikace, i když není implementována pomocí vlastních modelů v rámci aplikace `main`.



# Fáze 3 – Propojení URL → View → Template

## Část 1: Aplikace main

* 📍 URL cesty:
* - `/` → `home\_landing` → `main/home\_landing.html`
* - `/play\_redirect/` → `play\_redirect` → přesměrování dle přihlášení
* - `/<cokoliv>` → `fallback\_redirect` → přesměrování na hlavní stránku
* 🧠 View funkce:
* - `home\_landing(request)` – vykreslí hlavní stránku (šablona `main/home\_landing.html`)
* - `play\_redirect(request)` – kontroluje přihlášení a přesměrovává na hru nebo login
* - `register(request)` – zpracovává registraci uživatele (formulář `UserCreationForm`)
* - `fallback\_redirect(request, unused\_path)` – přesměrování na hlavní stránku při neexistující adrese

## Část 2: Aplikace gameplay

* 🔹 `/start/` → `start\_new\_game(request)`
* - Smaže všechny nedokončené hry přihlášeného hráče.  
  - Zjistí, zda je to první hra → aktivuje intro přes session.  
  - Získá obtížnost z parametru (?difficulty=...) nebo nastaví 'easy'.  
  - Vytvoří novou instanci `Game` a přesměruje na herní stránku.
* 🔹 `/<uuid>/` → `game\_view(request, game\_id)`
* - Zobrazí celou mřížku 9×9 buněk dané hry.  
  - Podle `block\_index` zvýrazní příslušný 3×3 blok.  
  - Zobrazí místnost, která odpovídá bloku.  
  - Načítá šablonu pro hlavní herní obrazovku.
* 🔹 `/place/<cell\_id>/` → `place\_item(request, cell\_id)`
* - Přijímá POST požadavek (AJAX) s `item\_id`, který hráč vybral.  
  - Zapíše vybranou položku (`selected\_item`) do buňky (`Cell`).  
  - Pokud je hra tímto dokončena:  
   – Označí hru jako dokončenou  
   – Spustí `try\_unlock\_memory()` pro přidání vzpomínky  
   – Vrátí JSON se sekvencí (`get\_sequence\_for\_trigger()`)
* 🔹 `/story/` → `story\_so\_far(request)`
* - Zobrazuje stránku s odemčeným příběhem hráče.  
  - Načte `Intro`, `Memory` i `DifficultyTransition` podle pokroku hráče.  
  - Rozhodne, kterou sekvenci přehrát (např. memory, hard\_end...)  
  - Předá šabloně `sequence.html` texty i obrázky pro přehrání příběhu.
* 🔹 `/game/` → `game\_selection(request)`
* - Hlavní stránka s výběrem hry (nová / rozehraná / nastavení obtížnosti).  
  - Zjišťuje, zda má hráč rozehranou hru (`Game`) nebo odemčené vzpomínky (`PlayerStoryProgress`).  
  - Pokud nemá žádnou rozehranou hru ani vzpomínku, přehraje se intro sekvence.  
  - Vrací šablonu `gameplay/game\_selection.html` s daty o postupu a sekvencemi.
* 🔹 `/manual/` → `manual\_view(request)`
* - Zobrazí stránku s návodem a pravidly hry.  
  - Vrací šablonu `gameplay/manual.html`.
* 🔹 `/auto\_fill/<uuid>/` → `auto\_fill(request, game\_id)`
* - Debug funkce: vyplní všechny buňky ve hře správně.  
  - Slouží pro testování výhry.
* 🔹 `/debug/reset\_progress/` → `reset\_progress(request)`
* - Debug funkce: smaže rozehrané hry, skóre i příběhový pokrok hráče.
* 🔹 `/debug/add\_memory/<difficulty>/` → `debug\_add\_memory(request, difficulty)`
* - Debug funkce: odemkne jednu vzpomínku v dané obtížnosti (easy/medium/hard).  
  - Užitečné pro testování přechodových sekvencí.

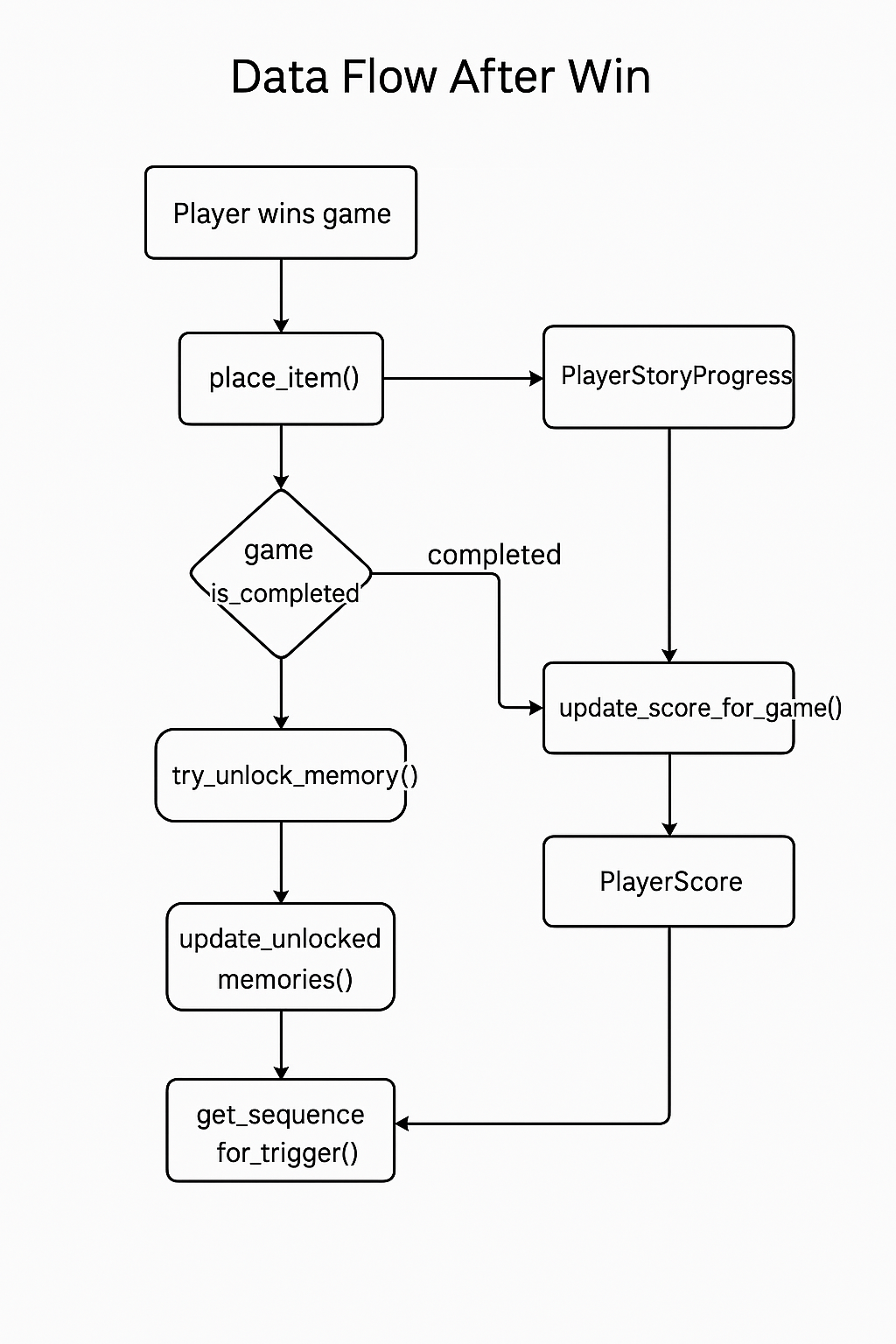
## Část 3: Aplikace score

* 🔹 `/score/` → `scoreboard(request)`
* - Zobrazí tabulku s výsledky všech hráčů (model `PlayerScore`).  
  - Lze řadit podle různých metrik (?sort=...):  
   – Počet her, nejlepší čas, počet vzpomínek  
  - Každému hráči se počítá:  
   – Pořadí v tabulce (rank)  
   – Procento odemčeného příběhu (vzpomínky / 60)
* 🔹 `/score/api/scoreboard/` → `api\_scoreboard(request)`
* - API endpoint vracející JSON s výsledky hráčů (`PlayerScore`).  
  - Podporuje parametry:  
   – `limit` (výchozí 100)  
   – `offset` (výchozí 0)  
  - Vrací seznam hráčů s těmito údaji:  
   – username, počty dokončených her, nejlepší časy, počet vzpomínek.
* 🔹 `/score/api/docs/` → `api\_docs(request)`
* - Načte dokumentaci API ze souboru `docs/api.md` ve formátu Markdown.  
  - Pomocí Python knihovny `markdown` převede text do HTML.  
  - Vykreslí výstup pomocí šablony `score/api\_docs.html`, která zobrazí dokumentaci v prohlížeči.

# Fáze 4 – Tok dat mezi částmi systému

## Část 1: Výhra hry → odměna

* `place\_item(request, cell\_id)`
* - Hráč odešle svůj výběr položky (AJAX POST)  
  - Zapsání vybrané položky do `Cell`  
  - Kontrola výhry (`game.is\_completed()`)  
  - Pokud hráč vyhrál:  
   – označí `Game.completed = True`  
   – zavolá `try\_unlock\_memory(game)`  
   – zavolá `update\_score\_for\_game(game)`  
   – smaže instanci `Game`
* `try\_unlock\_memory(game)`
* - Zjistí obtížnost a hráče  
  - Načte nebo vytvoří `PlayerStoryProgress`  
  - Vybere náhodnou neodemčenou `Memory` pro danou obtížnost  
  - Přidá ID vzpomínky do příslušného seznamu (`unlocked\_easy`, ...)  
  - Uloží změny a vrátí novou vzpomínku
* `update\_score\_for\_game(game)`
* - Získá nebo vytvoří záznam `PlayerScore`  
  - Zvýší počítadla dohraných her podle obtížnosti  
  - Zapíše aktuální čas a nejlepší dosažený čas  
  - Zavolá `update\_unlocked\_memories()` pro aktualizaci příběhu



## Část 2: Zápis skóre a přepočet příběhu

* `update\_score\_for\_game(game)`
* - Spočítá délku hry od vytvoření po výhru  
  - Načte nebo vytvoří `PlayerScore`  
  - Zvýší celkový počet dohraných her a počet podle obtížnosti  
  - Uloží první čas výhry a nejlepší čas  
  - Zavolá `update\_unlocked\_memories()` pro přepočet odemčených vzpomínek
* `update\_unlocked\_memories(self)`
* - Načte `PlayerStoryProgress` pro daného hráče  
  - Spočítá počet odemčených vzpomínek pro všechny obtížnosti  
  - Výsledek uloží do `PlayerScore.unlocked\_memories`  
  - Pokud progress neexistuje, nastaví hodnotu na 0

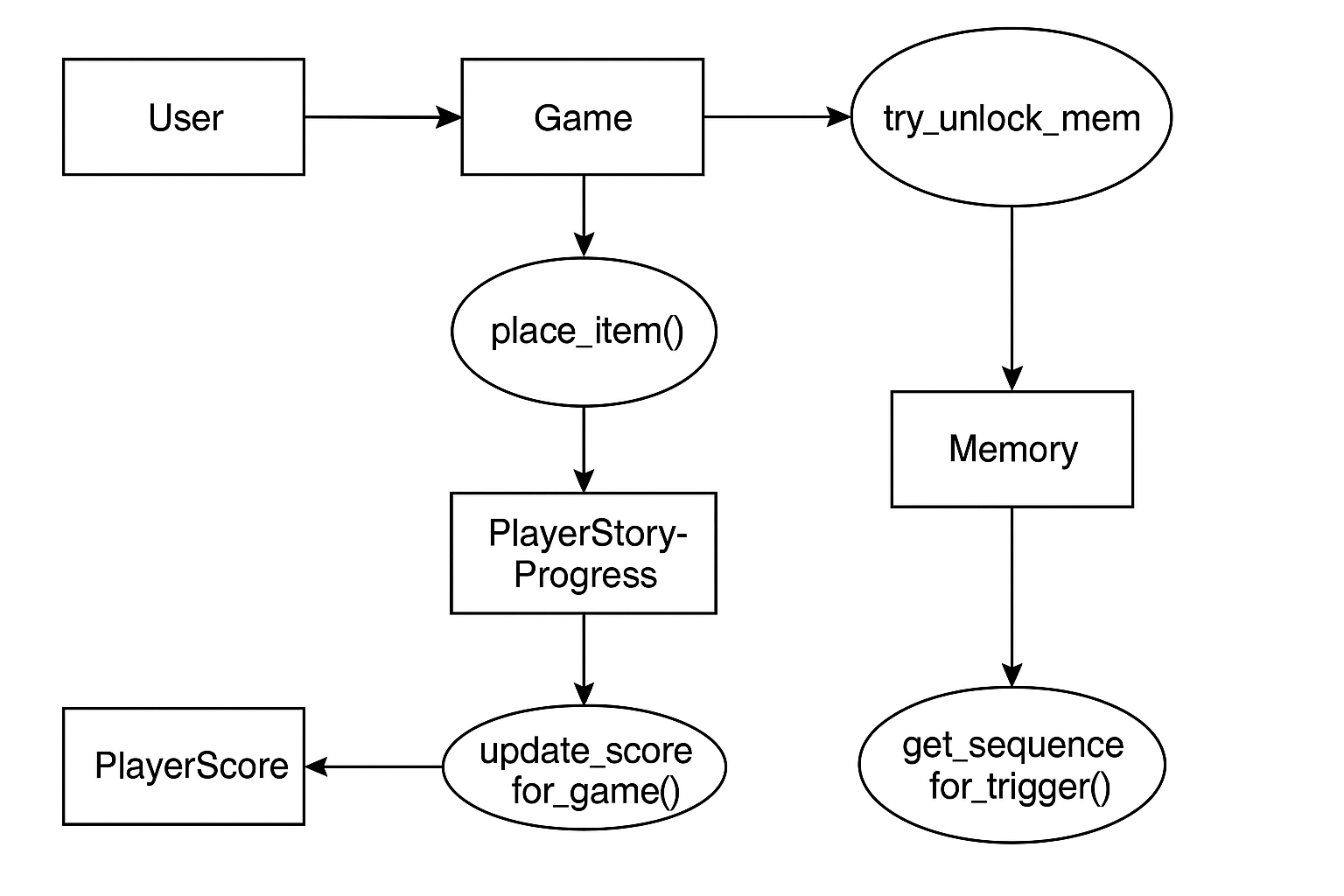
## Část 3: Výběr příběhové sekvence

* `get\_sequence\_for\_trigger(trigger, player, memory=None)`
* - Určuje, jaká příběhová sekvence se má přehrát po startu nebo výhře  
  - trigger = "start":  
   – Pokud hráč nemá žádné vzpomínky → přehraje se sekvence `intro`  
  - trigger = "complete":  
   – Pokud má 20 vzpomínek určité obtížnosti → `easy\_end`, `medium\_end`, `hard\_end`  
   – Pokud právě odemkl novou vzpomínku → přehraje se `memory`  
  - Pokud žádná podmínka neplatí → nevrací nic

## Část 4: Doplňky a související funkce

* `game.is\_completed()`
* - Funkce kontroluje, zda je \*\*všech 81 buněk správně vyplněno\*\*.  
   Pokud není, vrátí `False`, pokud ano, vrátí `True` a spustí logiku pro dokončení hry (uzavření).
* `reset\_progress()`
* - Funkce pro \*\*vymazání postupu hráče\*\*, jeho skóre a resetování všech herních dat. Používá se při testování nebo pokud hráč chce začít znovu.
* `Memory` objekty
* - Při \*\*odemykání vzpomínek\*\* se připojí příběhový text a další \*\*interaktivní prvky\*\*, které posouvají příběh vpřed.  
   Některé vzpomínky mohou mít \*\*dopad na další sekvence hry\*\* (např. změna obtížnosti, zpřístupnění nových výzev).
* Generování nové hry
* - Po dokončení \*\*aktuální hry\*\* a \*\*resetování postupu\*\* zavolá funkce `create\_game\_for\_player()` novou hru s \*\*novým nastavením mřížky\*\* a položek. Tento proces probíhá opakovaně po každé dokončené hře.

# Celkový obrázek Data Flow



## **Fáze 5 – Šablony a vykreslování**

## **Mystdoku:** base.html – základní šablona

* - Používá se jako základní šablona pro všechny stránky.
* - Obsahuje základní HTML strukturu, hlavičku, stylování a navigaci.
* Hlavní bloky:
* - `{% block title %}` – název stránky v `<title>`
* - `{% block content %}` – hlavní obsah v `<body>`
* Navigační prvky:
* - Zobrazení jména přihlášeného uživatele: `{% if user.is\_authenticated %}`
* - Odkaz na login/registraci pro nepřihlášené.
* - Odkazy na hlavní stránky: `HRAJ`, `SCORE`, `STORY`, `MANUAL`, atd.
* Styling a JS:
* - Načítá vlastní CSS styl a případně další skripty pro specifické podstránky.
* - Používá `{% block extra\_scripts %}` pro dynamické přidání JS.

## Aplikace: Gameplay

### Část 1: `game.html`

* `{% extends "base.html" %}` – šablona dědí základní layout
* Blok `{% block content %}` obsahuje hlavní herní obsah
* Inventář – `inventory-wrapper`
* - `for item in items` – vykresluje ikonky položek dostupných v místnosti  
  - Každý item má obrázek (podle `group\_id`) a název  
  - Mezi položky jsou vloženy prázdné sloty (pomocí `range9` a `used\_numbers`)
* Mřížka – `sudoku-grid`
* - `for cell in cells` – každá buňka odpovídá modelu `Cell`  
  - Pokud má `selected\_item`, vykreslí jeho číslo  
  - Pokud je hra v `easy` obtížnosti, zvýrazní správné / chybné odpovědi  
  - Střední buňka (`cell 40`) má obrázek jako pozadí
* Mini mapa – `room-minimap`
* - `for i in block\_range` – vykresluje 9 místností jako čtverce  
  - Aktivní blok má třídu `active`  
  - Uvnitř aktivní místnosti se vykresluje obrázek pozadí (např. `clock\_image.png`)
* Dveře mezi místnostmi – `block-with-doors`
* - `neighbors.up / left / right / down` – odkaz na sousední bloky  
  - `<a>` s akcí `onclick='transitionTo(...)'` – mění aktivní blok

### Část 2: `game\_selection.html`

* `{% extends "base.html" %}` – stránka dědí základní layout
* Rozehraná hra:
* - `{% if existing\_game %}` → zobrazuje tlačítko 'Pokračovat v hře'  
  - Přesměrování na `game\_view` pomocí UUID aktuální hry
* Nová hra podle obtížnosti:
* - Vždy se zobrazuje možnost `Easy`  
  - `Medium` se zobrazí, pokud hráč má 20 easy vzpomínek  
  - `Hard` se zobrazí, pokud hráč má 20 easy i 20 medium vzpomínek  
  - Tlačítka odkazují na `start\_new\_game` s parametrem `?difficulty=`
* Reset postupu:
* - Tlačítko pro reset příběhového postupu  
  - Volá view `reset\_progress`  
  - Potvrzovací dialog přes `onclick='confirm(...)'`
* Intro sekvence:
* - Na konci šablony se načítá `partials/sequence.html`  
  - Pokud je `play\_intro == True`, spustí se `playSequence('intro')` při načtení stránky

### Část 3: `story\_so\_far.html`

* `{% extends "base.html" %}` – stránka dědí základní layout
* Sekvence ke spuštění:
* - Tlačítka pro přehrání: `Intro`, `Easy End`, `Medium End`, `Hard End`  
  - Aktivují se podle pokroku hráče (20/20/60 vzpomínek)  
  - Volají `playSequence(...)` skrze JS
* Seznam odemčených vzpomínek:
* - Výpis všech dostupných vzpomínek (`memories`) ve formě citací  
  - Rozdělené podle obtížnosti (easy, medium, hard)  
  - Každá sekce je stylizovaná a přehledně zobrazená
* Konec hry a Credits:
* - Pokud má hráč všech 60 vzpomínek, zobrazí se sekce s 'Credits'  
  - Modal okno s poděkováním, jmény autorů a designem

### Část 4: `manual.html`

* `{% extends "base.html" %}` – stránka dědí základní layout
* Video s návodem:
* - Vložené YouTube video přes `<iframe>`  
  - Slouží jako úvod pro nové hráče
* Textový popis hry:
* - Vysvětlení pravidel MystDoku:  
   – V každé místnosti je 9 unikátních předmětů  
   – Nelze opakovat ve sloupci, řádku ani místnosti  
  - Hráč se přesouvá kliknutím na dveře

### Část 5: `sequence.html`

* Samostatná šablona pro přehrávání textových sekvencí (intro, memory, ending)
* Není samostatná stránka – vkládá se přes `{% include 'partials/sequence.html' %}`
* Obsah sekvence:
* - `<div id='sequence-overlay'>` překryje celou stránku černým pozadím  
  - `#sequence-text` – místo pro animovaný text sekvence  
  - `.sequence-background.bg-a / bg-b` – přechody pozadí  
  - `#sequence-hint` – nápověda pro přeskok pomocí ESC
* Funkce a logika:
* - Sekvence se spouští voláním `playSequence(název)` v JavaScriptu  
  - Texty a obrázky jsou předány jako JS proměnné z contextu (např. `intro\_texts`)  
  - Umožňuje přeskok stiskem klávesy ESC  
  - Používá `setTimeout` pro animaci jednotlivých kroků

## Aplikace: main

#### Část 1: home\_landing.html

* - `{% extends 'base.html' %}` – základní layout
* - Zobrazuje tlačítko 'Hraj' a uvítací text
* - `{% if user.is\_authenticated %}` – přepíná mezi přihlášeným a nepřihlášeným režimem
* - Odkazy na přihlášení / registraci nebo na výběr hry

#### Část 2: login.html

* - `{% extends 'base.html' %}` – základní layout
* - Obsahuje formulář pro přihlášení uživatele
* - Pole: uživatelské jméno a heslo
* - Po odeslání: validace a přesměrování na `game\_selection`
* - Zobrazení chybové hlášky při neúspěchu

#### Část 3: register.html

* - `{% extends 'base.html' %}` – základní layout
* - Formulář pro registraci nového uživatele
* - Pole: jméno, heslo, potvrzení hesla
* - Po validaci vytvoří nového uživatele a přesměruje na login

## Aplikace: score

#### Část 1: scoreboard.html

* - `{% extends 'base.html' %}` – základní layout
* - Obsahuje tabulku všech hráčů (`for score in player\_scores`)
* - Sloupce: uživatel, celkem her, best časy, počet vzpomínek
* - Odkazy pro třídění tabulky: `?sort=username`, `?sort=unlocked\_memories`, atd.
* - Zvýraznění aktuálního hráče: `{% if user == score.user %}`
* - Stránkování výsledků: `page\_obj` + navigace

#### Část 2: api\_docs.html

* - `{% extends 'base.html' %}` – základní layout
* - Šablona pro zobrazení API dokumentace ve formátu HTML
* - Využívá proměnnou `api\_html`, která se vkládá pomocí `{{ api\_html|safe }}`
* - Obsah je převzatý z `docs/api.md` a převedený knihovnou `markdown`

## Fáze 6 – Herní logika a pravidla

### Část 1: Kontrola výhry

* `Game.is\_completed()`
* - Zjistí, zda je hra úspěšně dokončena  
  - Načte všechny buňky (`Cell`) patřící do dané hry  
  - Pro každou buňku kontroluje:  
   – zda má vybraný item (`selected\_item`)  
   – zda je odpověď správná (`is\_correct()`)  
  - Pokud jakákoli buňka není správně vyplněna → hra není dokončena  
  - Pokud všechny buňky jsou správně → vrací `True`
* `Cell.is\_correct()`
* - Ověří, zda je odpověď v buňce správná  
  - Vrací `True`, pokud:  
   – Hráč vybral nějaký item  
   – Buňka má správný item (`correct\_item`)  
   – Číslo obou položek (`.number`) je shodné  
  - Díky tomu je možné používat různé názvy (synonyma) se stejným číslem

### Část 2: Rozdíly podle obtížnosti

* Počet předvyplněných položek:
* - `generate\_game(difficulty)` určuje obtížnost při startu nové hry  
  - Rozdílný počet buněk je automaticky vyplněn správně (`selected\_item = correct\_item`)  
  - `easy`: mnoho předvyplněných buněk  
  - `medium`: střední obtížnost  
  - `hard`: téměř žádná pomoc, většina mřížky prázdná
* Zobrazení správnosti odpovědí:
* - Pouze u obtížnosti `easy` se v šabloně `game.html` barví buňky  
  - CSS třídy `correct` a `wrong` se přidávají dle `is\_correct()`  
  - Hráč vidí, zda odpověděl správně / špatně

### Část 3: Generování nové hry

* `create\_game\_for\_player(player, difficulty)`
* - Generuje kompletní hru založenou na principu Sudoku  
  - 1) Vygeneruje klasickou Sudoku mřížku 9×9 s čísly 1–9 (`generate\_sudoku()`)  
  - 2) Vybere 9 místností (`select\_valid\_rooms()`), každá s 9 unikátními položkami  
  - 3) Pro každou místnost vytvoří mapování: blok → 9 položek (`block\_items`)  
  - 4) Vytvoří objekt `Game` s přiřazenými místnostmi (`block\_rooms`) a položkami  
  - 5) Vytvoří jednotlivé buňky (`Cell`) podle číselné mřížky a zvolené obtížnosti (`fill\_cells()`)  
  - 6) Vrací hru pouze pokud je výsledná kombinace řešitelná (`is\_sudoku\_solvable()`)

### Část 4: Vytváření buněk pomocí `fill\_cells()`

* `fill\_cells(game, board, block\_items, difficulty)`
* - Vytváří všech 81 objektů `Cell` pro zadanou hru  
  - 1) Nastaví počet buněk, které mají být viditelné (předvyplněné):  
   – `easy`: 36  
   – `medium`: 30  
   – `hard`: 24  
  - 2) Náhodně určí, které buňky zůstanou prázdné  
  - 3) Pro každou buňku v 9×9 mřížce:  
   – Určí její 3×3 blok (0–8)  
   – Vybere správný item podle `block\_items` a čísla z mřížky  
   – Vytvoří objekt `Cell` se správným a případně zvoleným itemem  
   – Nastaví `prefilled=True/False` dle výběru

### Část 5: Validace řešitelnosti

* `is\_sudoku\_solvable(board)`
* - Funkce ověřuje, zda je náhodně vygenerovaná sudoku mřížka logicky řešitelná.  
  - Pokud mřížka nelze logicky dokončit, hra se nevytvoří a pokus se opakuje.  
  - Zajišťuje férovost a logiku výsledné hry.

### Část 6: Koordináty buněk a logika bloků

* Generování `Cell`
* - Každá buňka má přiřazené souřadnice `x` a `y` podle pozice v mřížce (0–8).  
  - Pomocí těchto souřadnic se vypočítá blok (0–8), ke kterému buňka patří.  
  - Logika bloku: `(y // 3) \* 3 + (x // 3)`.  
  - Tato informace je klíčová pro správné rozdělení místností v herní mřížce.

### Část 7: Výhra a odstranění hry

* Dokončení hry
* - Po dokončení hry (`game.is\_completed() == True`) se spouští proces odměny.  
  - Volají se funkce `try\_unlock\_memory(game)` a `update\_score\_for\_game(game)`.  
  - Po zapsání dat se instance `Game` smaže – zůstává pouze záznam o skóre a vzpomínce.

## Fáze 7 – Přístupová práva a zabezpečení

### 1. Přihlášení a registrace

* - Projekt používá vestavěný Django `User` model  
  - Uživatelské rozhraní pro přihlášení a registraci je v aplikaci `main`  
  - View funkce `login\_view` a `register\_view` zajišťují přihlášení a vytvoření nového účtu  
  - Po přihlášení je uživatel dostupný přes `request.user`

### 2. Ochrana přístupu ke hře

* - Hráč musí být přihlášen pro přístup ke hře, výběru, příběhu i skóre  
  - V jednotlivých view funkcích se používá podmínka:  
   `if not request.user.is\_authenticated: return redirect('login')`  
  - Zároveň je použito `@login\_required`, jako další explicitní kontrola

### 3. Chování pro nepřihlášené hráče

* - Při pokusu o vstup na chráněné view je uživatel přesměrován na přihlašovací stránku  
  - Úvodní stránka (`home\_landing.html`) zobrazuje různé prvky podle stavu přihlášení  
  - Pokud není přihlášen: `LOGIN / REGISTRACE`  
  - Pokud je přihlášen: jméno hráče vpravo nahoře + odkazy na hru

### 4. Admin sekce

* - V době vydání nebyla implementována samostatná administrační sekce  
  - Úpravy dat (`items.json`, `story.json`) jsou prováděny ručně mimo runtime  
  - Budoucí verze může obsahovat dedikovaný editor a validaci

## Fáze 8 – Validace a ladicí funkce

📌 Upozornění: Validace souborů `items.json` a `story.json` byla plánována, ale nebyla implementována v rámci vydané verze.

Projekt používá ověřené, ručně zkontrolované verze těchto souborů jako oficiální zdroj herních dat.

### Ladicí a testovací view funkce

* `/auto\_fill/<uuid>/` → `auto\_fill(request, game\_id)`
* - Vyplní herní mřížku správnými odpověďmi automaticky  
  - Slouží pro testování výhry a navazujících přechodových sekvencí
* `/debug/reset\_progress/` → `reset\_progress(request)`
* - Smaže hráčův postup včetně rozehraných her, skóre a příběhu  
  - Používáno při vývoji pro rychlé restartování testovacího účtu
* `/debug/add\_memory/<difficulty>/` → `debug\_add\_memory(request, difficulty)`
* - Přidá jednu náhodnou vzpomínku hráči pro danou obtížnost  
  - Používá se pro testování přehrávání `memory`, `easy\_end`, `hard\_end`, atd.

### Debug blok ve `story\_so\_far.html`

* V šabloně `story\_so\_far.html` je zakomentovaný rozšířený ladicí režim (`{% comment %} ... {% endcomment %}`)
* Co obsahuje ladicí blok:
* - Tlačítka pro ruční přehrání sekvencí: `intro`, `easy\_end`, `medium\_end`, `hard\_end`  
  - Výpis všech načtených řádků z `sequences[...]` pro kontrolu textových dat  
  - Přehled všech odemčených vzpomínek podle obtížnosti (`mem.order`, `mem.text`)  
  - Debug odkazy na přidání vzpomínek pomocí `debug\_add\_memory`  
  - Odkaz na `reset\_progress` pro vymazání všech dat
* Tento blok sloužil jako vývojářská konzole pro kontrolu toku příběhu a testování koncových sekvencí.

# Závěr a možnosti rozšíření

## Shrnutí projektu MystDoku

Projekt MystDoku je originální kombinací klasického sudoku a příběhové logiky, ve které hráč postupně odemyká vzpomínky prostřednictvím dokončených her. Celý systém je navržen jako webová aplikace s rozdělením na více Django aplikací, kde každá plní specifickou roli – od správy uživatelů, přes herní logiku, až po sledování skóre a postupu.

Dokumentace pokrývá veškeré klíčové části systému – datové modely, šablony, funkce ve view, ladicí nástroje i API. Zároveň je pečlivě ověřena proti skutečné implementaci a slouží jako kvalitní referenční zdroj pro další vývoj a správu.

## Možnosti rozšíření a další vývoj

* Administrace dat:
* - Vytvořit webové rozhraní pro správu položek (`items.json`), vzpomínek (`story.json`) a sekvencí (`sequence\_frames.json`).
* - Validace a vizuální nástroje pro kontrolu duplicit, počtů a přehlednosti.
* Pokročilé debugování a testování:
* - Implementace funkce `validate\_game\_data()` pro kontrolu integrity každé hry.
* - Záznam ladicích výstupů a logování nevalidních stavů hry.
* Uživatelské funkce a gamifikace:
* - Úrovně hráčů, odměny za pokrok, odemykání nových motivů a skinů.
* - Statistiky hraní, denní výzvy nebo náhodné výběry příběhů.
* Lokalizace a jazykové verze:
* - Podpora vícejazyčných překladů pro širší publikum.
* Podpora více rozlišení
* Podpora mobilní verze