ChatGPT

Supa-crawl (LLM-playground) – architecture & workflow

Overview

The `supa-crawl` project is a Python web-crawling pipeline that wraps the Crawl4AI framework with Supabase storage. The `AdvancedWebCrawler` class configures a headless Chromium browser (via `BrowserConfig`) and creates a `CrawlerRunConfig` for each crawl [1]. It supports memory-adaptive and semaphore dispatchers for multi-URL crawling [2] and integrates an LLM extraction strategy to produce JSON summaries [3]. Results (URL, raw markdown, analysis) can be stored in a Supabase `pages` table using a dedicated handler [4]. `main.py` orchestrates tests of each dispatcher and the LLM pipeline [5]. Environment variables specify the Supabase URL/key and OpenAI API key [6]. Official Crawl4AI docs explain why LLM extraction is used (for unstructured or semantic data) [7] and how dispatchers offer adaptive crawling with rate limiting [8]. Supabase CLI docs describe how to initialize and start a local Supabase project with `supabase init` and `supabase start` [9] [10].

Sources used

Official documentation and code were referenced from:

- **Crawl4AI docs** – LLM extraction strategy, dispatcher behaviour and configuration [7] [8] [11].
- **Supabase docs** – CLI commands (`supabase init`, `start`, `stop`) for local development [9] [10] [12].
- **Repository code** – `src/config/environment.py`, `src/crawlers/async_crawler.py`, `src/storage/supabase_handler.py`, `src/models/schemas.py` and `main.py` on the `LLM-playground` branch [6] [1] [4] [13] [14].

Directory tree (relevant parts)

```
supa-crawl/
├─ src/
│  ├─ config/
│  │  └─ environment.py          # loads env vars and creates browser/crawler
config
│  ├─ crawlers/
│  │  └─ async_crawler.py        # defines AdvancedWebCrawler and crawl methods
│  ├─ models/
│  │  └─ schemas.py              # Pydantic model for LLM extraction
│  └─ storage/
│     └─ supabase_handler.py     # handles Supabase client and insertion/upsert
```

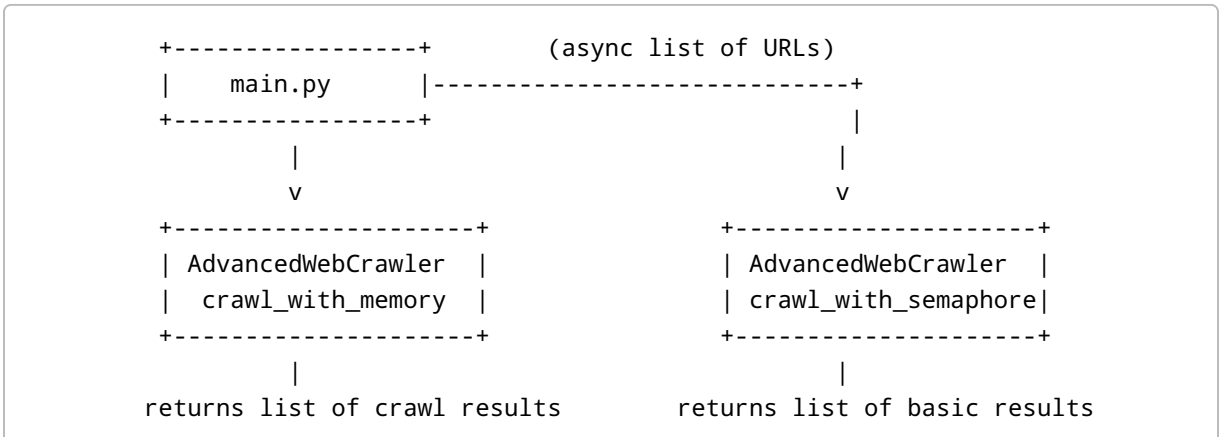
```
├─ main.py                          # demo entry point testing each pipeline stage
├─ supabase/                        # migrations and config for Supabase project
├─ docs/                            # documentation (not analysed)
├─ supabase-cli-commands.md         # local guide for Supabase CLI commands
├─ tests/                           # pytest scripts
└─ ... (other packaging files)
```

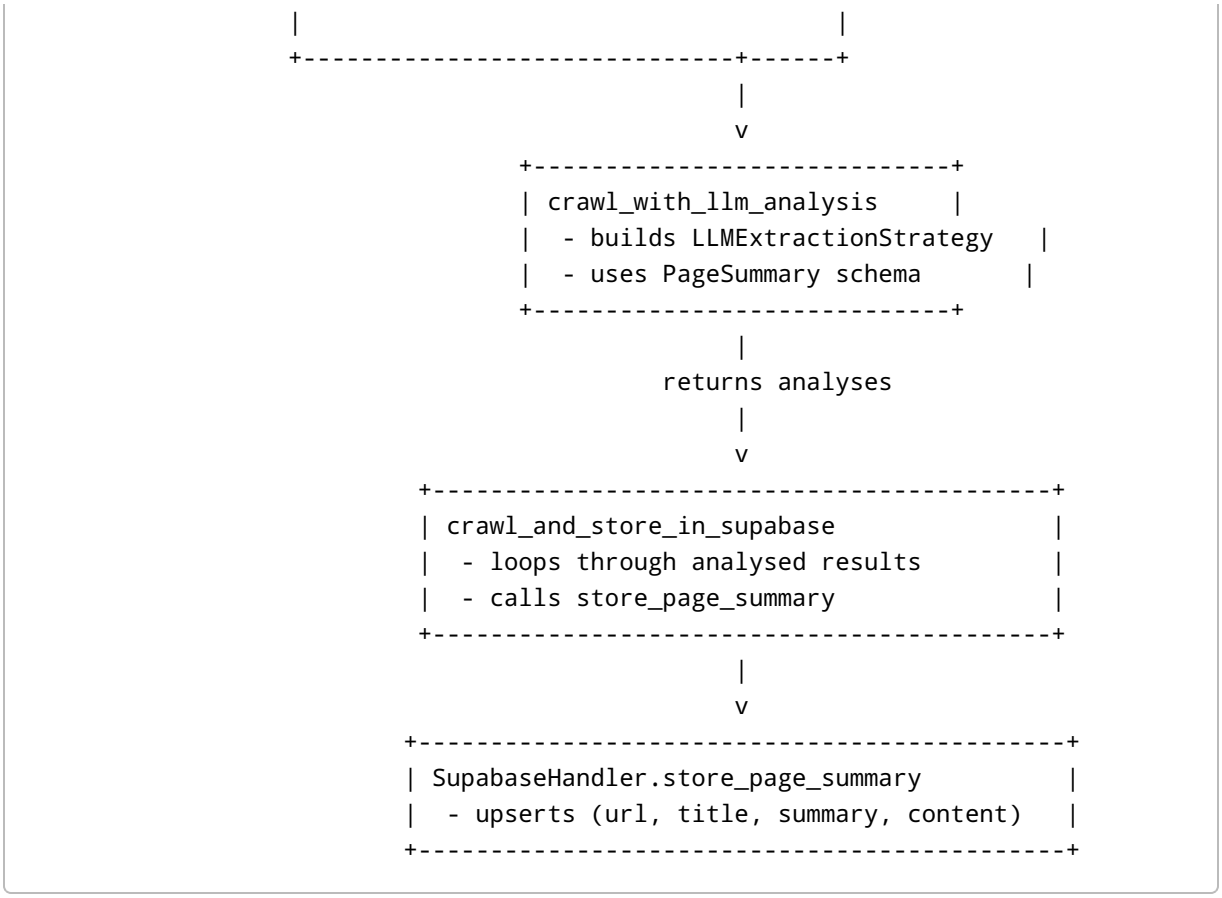
File breakdown

Path	Role	Key functions/classes	Called by / Call
`src/config/environment.py`	Centralised environment and crawler configuration	`EnvironmentConfig` reads `SUPABASE_URL`, `SUPABASE_KEY` and `OPENAI_API_KEY` from `.env` [6]; `CrawlerConfig.create_browser_config` builds a `BrowserConfig` with stealth settings [15]; `create_crawler_run_config` returns a `CrawlerRunConfig` with default extraction strategy [16]	Imported by `a` for global conf
`src/models/schemas.py`	Pydantic schemas for LLM extraction	`PageSummary` defines `title` and `summary` fields [18]; `CrawlResult` and `Crawl4AIResponse` wrap results [19]	Used by `asyn` configuring `LLMExtracti`
`src/crawlers/async_crawler.py`	Implements `AdvancedWebCrawler`	Constructor sets `BrowserConfig` and `CrawlerRunConfig` [1] and initialises `SupabaseHandler` [21]. `crawl_with_memory_adaptive_dispatcher` configures a `MemoryAdaptiveDispatcher` with rate limiting and memory thresholds [2]; `crawl_with_semaphore_dispatcher` uses `SemaphoreDispatcher` for simple concurrency [22]. `crawl_with_llm_analysis` checks for an OpenAI key and builds an `LLMExtractionStrategy` with a multi-line prompt, schema and chunking parameters [3]; it parses JSON results and returns analysis dictionaries [23]. `crawl_and_store_in_supabase` calls the LLM crawl and stores each summary in Supabase via `store_page_summary` [24].	Called by `mai` demonstration `SupabaseHan` `PageSummary`

Path	Role	Key functions/classes	Called by / Call
`src/storage/supabase_handler.py`	Wraps Supabase client and storage operations	`_initialize_client` creates a Supabase client when credentials exist [25]. `_extract_first_paragraph` trims markdown to the first meaningful paragraph for storage [26]. `store_crawl_results` inserts `url` and a truncated `content` into the `pages` table; optionally prepends LLM title/summary [27]. `store_page_summary` performs an upsert of `url`, `title`, `summary` and optional `content` using Supabase-py's `upsert()` [28].	Called by `asyn` (`crawl_and_s` and tests
`main.py`	Example script demonstrating the crawler	Defines `urls` list of target pages [29]. Inside `main()` it initialises `AdvancedWebCrawler`, then sequentially tests memory-adaptive crawling, semaphore dispatching, LLM analysis and full pipeline storage [5]	Run directly via
`supabase-cli-commands.md`	Local CLI reference (not official but summarises Supabase CLI usage)	Contains commands for installation, authentication, migrations, data inspection and troubleshooting	Stand-alone; us onboarding

Workflow diagram





Configuration & environment variables

- `.env` – must define `SUPABASE_URL`, `SUPABASE_KEY` and `OPENAI_API_KEY`. `EnvironmentConfig` loads these and warns if missing [30].
- **Crawler config** – `BrowserConfig` parameters (headless, browser type, user agent) and `CrawlerRunConfig` settings (cache mode, word count, extraction strategy) follow official patterns [1].
- **Dispatchers** – choose between `MemoryAdaptiveDispatcher` (pauses based on RAM usage; includes rate limiter and monitor) and `SemaphoreDispatcher` (fixed concurrency). Crawl4AI docs explain that dispatchers provide adaptive crawling, rate limiting and real-time monitoring [8].
- **LLM extraction** – define a `PageSummary` schema and specify `LLMExtractionStrategy` parameters like provider, API token, instruction, chunking and extraction type [3]. Crawl4AI docs note that LLM extraction supports various providers and uses chunking to manage token limits [7] [31].
- **Supabase client** – initialised via `create_client(SUPABASE_URL, SUPABASE_KEY)` [25]. To run Supabase locally, official CLI commands are: `supabase init` to set up a project and `supabase start` to launch Postgres, Auth, Storage and Studio [9] [10]; `supabase stop` stops the services [12].

Limitations & next actions

- **Manual URL list** – `main.py` hard-codes test URLs [29]. For production use, implement URL ingestion (e.g., read from a file or database).
- **Schema rigidity** – `PageSummary` contains only `title` and `summary` [18]. To extract specific fields (e.g., prices or code snippets), define a new Pydantic schema and update the `LLMExtractionStrategy` prompt accordingly.
- **Error handling** – exceptions in `async_crawler.py` are printed but not retried beyond the dispatcher's rate limiter. Adding retry logic or logging would improve robustness.
- **Supabase table schema** – the `pages` table is assumed to have `url`, `title`, `summary` and `content` columns. Migrations aren't included; the `supabase` directory likely contains SQL files to create this schema but was not analysed.
- **Client dependencies** – the code expects the `supabase` Python client to be installed. `supabase_handler.py` exits if it cannot import `create_client` [32].

Next steps:

1. **Automate ingestion** – integrate a queue or scheduler to supply URLs for crawling, rather than a static list.
2. **Custom extraction** – extend `schemas.py` and modify the LLM prompt to pull domain-specific data (e.g., product prices, code examples).
3. **Database migrations** – ensure migrations in `supabase` folder align with the fields used in `store_page_summary`; document how to run them (`supabase db push` and `supabase migration create`).
4. **Monitoring & logging** – integrate logging libraries and consider using the Crawl4AI monitor API for real-time dashboards.
5. **Cost optimisation** – since LLM extraction can be slow and costly [7], implement a fallback to schema-based extraction when structured data is available.

---

1 2 3 20 21 22 23 24 async_crawler.py
https://github.com/Gaya56/supa-crawl/blob/LLM-playground/src/crawlers/async_crawler.py

4 25 26 27 28 32 supabase_handler.py
https://github.com/Gaya56/supa-crawl/blob/LLM-playground/src/storage/supabase_handler.py

5 14 29 main.py
https://github.com/Gaya56/supa-crawl/blob/LLM-playground/main.py

6 15 16 17 30 environment.py
https://github.com/Gaya56/supa-crawl/blob/LLM-playground/src/config/environment.py

7 31 LLM Strategies - Crawl4AI Documentation (v0.7.x)
https://docs.crawl4ai.com/extraction/llm-strategies/

8 Multi-URL Crawling - Crawl4AI Documentation (v0.7.x)
https://docs.crawl4ai.com/advanced/multi-url-crawling/

9 10 12 Supabase CLI | Supabase Docs
https://supabase.com/docs/guides/local-development/cli/getting-started

11 Browser, Crawler & LLM Config - Crawl4AI Documentation (v0.7.x)
https://docs.crawl4ai.com/core/browser-crawler-config/

13 18 19 schemas.py
https://github.com/Gaya56/supa-crawl/blob/LLM-playground/src/models/schemas.py