全体像（PHP/Laravel 版）

- フロント：そのまま React（/、/board、/card/:id）
- バックエンド：Laravel 11（`apps/server-php`）
- 非同期ジョブ：Laravel Queue（database ドライバ）＋ Supervisor
    - OCR：`thiagoalessio/tesseract_ocr` 経由で `tesseract` CLI を実行
    - PDF：`Imagick` or `pdftoppm` で画像化 → OCR
- 永続化：まずは **JSON** ファイル（将来 DB 移行しやすいよう Repository 抽象化）
    - `storage/app/data/cards.json`, `events.json`, `jobs.json`, `documents.json`
- アップロード：`storage/app/uploads`（10MB 制限）
- 対応言語：`eng`, `jpn`, `jpn_vert`（OS に学習データ配置）
- **CORS**：React からのアクセス許可
- 実行：`php artisan serve` / `php artisan queue:work`

ディレクトリ構成（提案）

```
repo-root/
  apps/
    server-php/
      app/
        DTO/
          CardRecord.php
          CardSummary.php
          CardDetailOutputs.php
          ReactionSummary.php
          DepartmentRatio.php
        Http/Controllers/
```

```
    DocumentController.php
    JobController.php
    CardController.php
  Jobs/
    OcrProcessJob.php
  Repositories/
    CardRepository.php
    JobRepository.php
    DocumentRepository.php
  Services/
    OcrService.php
    PdfService.php
    WebClipService.php
bootstrap/
config/
  filesystems.php
  cors.php
  queue.php
database/
  migrations/
    2025_..._create_jobs_table.php     // queue:table 用
routes/
  api.php
storage/
  app/
    uploads/
    data/
      cards.json
      events.json
      jobs.json
      documents.json
composer.json
.env
supervisor.conf.sample
```

API（互換エンドポイント）

- ドキュメント・ジョブ

    - `POST /v1/documents` … ファイル or URL 受け取り → 非同期 **OCR** ジョブ発行
    - `GET /v1/jobs/{jobId}` … `{status, progress, document_id}`
    - `GET /v1/documents/{documentId}/text` … 抽出テキスト取得
- カード
    - `GET /api/board` … ボード一覧（要約配列）
    - `GET /api/cards/{id}` … 詳細
    - `PUT /api/cards/{id}` … 編集（保存＝永続化）
    - `DELETE /api/cards/{id}` … 削除（保存＝永続化）

既存フロント（ポーリング 500ms/30s、ローディング UI など）はそのまま利用可能。

主要実装（抜粋）

1) composer 依存

```
{
  "require": {
    "php": "^8.2",
    "laravel/framework": "^11.0",
    "thiagoalessio/tesseract_ocr": "^2.13",
    "symfony/dom-crawler": "^7.0",
    "symfony/css-selector": "^7.0"
  }
}
```

サーバに `tesseract`、（PDF→画像化用に）`imagemagick` または `poppler-utils` をインストール。

2) DTO（例：CardRecord）

```
// app/DTO/CardRecord.php
```

```php
namespace App\DTO;

class CardRecord {
    public string $id;
    public CardSummary $summary;
    public CardDetailOutputs $detail;
    public ReactionSummary $reactions;
    /** @var
array<int,array{date:string,views:int,comments:int,likes:int}> */
    public array $timeseries = [];
    /** @var DepartmentRatio[] */
    public array $audience = [];
}
```

（CardSummary, CardDetailOutputs, ReactionSummary, DepartmentRatio も同様の POPO で定義）

3) JSON リポジトリ（例：CardRepository）

```php
// app/Repositories/CardRepository.php
namespace App\Repositories;

use Illuminate\Support\Facades\Storage;

class CardRepository {
    private string $path = 'data/cards.json';

    /** @return array<string,mixed> */
    private function read(): array {
        if (!Storage::disk('local')->exists($this->path)) return [];
        $json = Storage::disk('local')->get($this->path);
        return $json ? json_decode($json, true) : [];
    }

    /** @param array<string,mixed> $data */
    private function write(array $data): void {
        Storage::disk('local')->put($this->path, json_encode($data,
JSON_PRETTY_PRINT|JSON_UNESCAPED_UNICODE));
    }

    /** @return array<int,array<string,mixed>> */
```

```
    public function listSummaries(): array {
        $all = $this->read();
        return array_map(fn($r) => $r['summary'] ?? [],
array_values($all));
    }

    /** @return array<string,mixed>|null */
    public function find(string $id): ?array {
        $all = $this->read();
        return $all[$id] ?? null;
    }

    /** @param array<string,mixed> $record */
    public function upsert(string $id, array $record): void {
        $all = $this->read();
        $all[$id] = $record;
        $this->write($all);
    }

    public function delete(string $id): void {
        $all = $this->read();
        unset($all[$id]);
        $this->write($all);
    }
}
```

`JobRepository`, `DocumentRepository` も同様に `jobs.json`, `documents.json` を管理（`status: queued|running|succeeded|failed`, `progress: 0-100`, `document_id`, `text`, `source_type: upload|url` などを保持）。

4) ルーティング

```
// routes/api.php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\{DocumentController, JobController,
CardController};

Route::post('/v1/documents', [DocumentController::class, 'store']);
Route::get ('/v1/jobs/{id}', [JobController::class, 'show']);
```

```
Route::get ('/v1/documents/{id}/text', [DocumentController::class,
'text']);

Route::get   ('/api/board', [CardController::class, 'board']);
Route::get   ('/api/cards/{id}', [CardController::class, 'show']);
Route::put   ('/api/cards/{id}', [CardController::class, 'update']);
Route::delete('/api/cards/{id}', [CardController::class,
'destroy']);
```

5) ドキュメント登録 → ジョブ発行

```
// app/Http/Controllers/DocumentController.php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Str;
use Illuminate\Support\Facades\Storage;
use App\Repositories\{DocumentRepository, JobRepository};
use App\Jobs\OcrProcessJob;

class DocumentController extends Controller {
    public function store(Request $req, DocumentRepository $docs,
JobRepository $jobs) {
        $validated = $req->validate([
            'file' =>
'nullable|file|mimetypes:image/jpeg,image/png,image/gif,application/
pdf,text/plain|max:10240',
            'url'  => 'nullable|url',
            'lang' => 'nullable|string' // e.g. "jpn+eng"
        ]);
        if (!$req->hasFile('file') && !$validated['url'] ?? null) {
            return response()->json(['error' => 'file or url
required'], 422);
        }

        $documentId = (string) Str::uuid();
        $sourceType = $req->hasFile('file') ? 'upload' : 'url';
        $storedPath = null;

        if ($sourceType === 'upload') {
```

```
            $storedPath =
$req->file('file')->store("uploads/{$documentId}", 'local');
        } else {
            $content = file_get_contents($validated['url']);
            $ext = 'html';
            $storedPath = "uploads/{$documentId}/remote.$ext";
            Storage::disk('local')->put($storedPath, $content);
        }

        $docs->upsert($documentId, [
            'id' => $documentId,
            'source_type' => $sourceType,
            'path' => $storedPath,
            'url' => $validated['url'] ?? null,
            'lang' => $validated['lang'] ?? 'jpn+eng',
            'text' => null,
            'created_at' => now()->toIso8601String(),
        ]);

        $jobId = (string) Str::uuid();
        $jobs->upsert($jobId, [
            'id' => $jobId,
            'status' => 'queued',
            'progress' => 0,
            'document_id' => $documentId,
            'created_at' => now()->toIso8601String(),
        ]);

        OcrProcessJob::dispatch($jobId);

        return response()->json([
            'job_id' => $jobId,
            'document_id' => $documentId,
        ]);
    }

    public function text(string $id, DocumentRepository $docs) {
        $doc = $docs->find($id);
        if (!$doc) return response()->json(['error' => 'not found'],
404);
        return response()->json(['text' => $doc['text'] ?? '']);
```

```
    }
}
```

6) ジョブ進捗（ポーリング応答）

```
// app/Http/Controllers/JobController.php
namespace App\Http\Controllers;

use App\Repositories\JobRepository;

class JobController extends Controller {
    public function show(string $id, JobRepository $jobs) {
        $job = $jobs->find($id);
        if (!$job) return response()->json(['error' => 'not found'],
404);
        return response()->json([
            'status' => $job['status'],
            'progress' => $job['progress'],
            'document_id' => $job['document_id']
        ]);
    }
}
```

7) OCR ジョブ本体

```
// app/Jobs/OcrProcessJob.php
namespace App\Jobs;

use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use App\Repositories\{JobRepository, DocumentRepository};
use App\Services\{OcrService, PdfService, WebClipService};

class OcrProcessJob implements ShouldQueue {
    use Queueable;

    public function __construct(private string $jobId) {}

    public function handle(JobRepository $jobs, DocumentRepository
$docs, OcrService $ocr, PdfService $pdf, WebClipService $clip) {
```

```
        $job = $jobs->find($this->jobId);
        if (!$job) return;

        $jobs->updateStatus($this->jobId, 'running', 1);
        $docId = $job['document_id'];
        $doc   = $docs->find($docId);

        try {
            $text = '';
            if ($doc['source_type'] === 'upload') {
                $path = storage_path("app/{$doc['path']}");
                if (preg_match('/\.pdf$/i', $path)) {
                    $pages = $pdf->pdfToImages($path); // array of
image paths
                    $total = max(count($pages), 1);
                    $i = 0;
                    foreach ($pages as $img) {
                        $text .= $ocr->imageToText($img,
$doc['lang']);
                        $i++;
                        $jobs->updateProgress($this->jobId,
intval($i/$total*98));
                    }
                } else {
                    $text = $ocr->imageOrTextToText($path,
$doc['lang']);
                    $jobs->updateProgress($this->jobId, 90);
                }
            } else {
                // URL → Web クリップ → テキスト抽出（見出し、本文ブロック
抽出）
                $rawPath = storage_path("app/{$doc['path']}");
                $html = file_get_contents($rawPath);
                $text = $clip->extractMainText($html);
                $jobs->updateProgress($this->jobId, 90);
            }

            $docs->updateText($docId, $text);
            $jobs->updateStatus($this->jobId, 'succeeded', 100);
        } catch (\Throwable $e) {
            $jobs->fail($this->jobId, $e->getMessage());
```

```
        }
    }
}
```

OCR/PDF/クリップのサービス例

```
// app/Services/OcrService.php
namespace App\Services;
use thiagoalessio\TesseractOCR\TesseractOCR;

class OcrService {
    public function imageToText(string $imagePath, string
$lang='jpn+eng'): string {
        return (new TesseractOCR($imagePath))
            ->lang(...explode('+', $lang))
            ->oem(1)->psm(3)
            ->run();
    }
    public function imageOrTextToText(string $path, string $lang):
string {
        if (preg_match('/\.txt$/i', $path)) return
file_get_contents($path) ?: '';
        return $this->imageToText($path, $lang);
    }
}

// app/Services/PdfService.php
namespace App\Services;

class PdfService {
    /** @return string[] image paths */
    public function pdfToImages(string $pdfPath): array {
        // Imagick があればこちら
        if (extension_loaded('imagick')) {
            $im = new \Imagick();
            $im->setResolution(300, 300);
            $im->readImage($pdfPath);
            $out = [];
            foreach ($im as $i => $page) {
                $page->setImageFormat('png');
```

```
                $tmp =
sys_get_temp_dir()."/ocr_page_{$i}_".uniqid().".png";
                $page->writeImage($tmp);
                $out[] = $tmp;
            }
            $im->clear(); $im->destroy();
            return $out;
        }
        // なければ pdftoppm
        $dir = sys_get_temp_dir()."/ocr_".uniqid();
        @mkdir($dir);
        $base = $dir.'/page';
        $cmd  = sprintf('pdftoppm -png -r 300 %s %s',
escapeshellarg($pdfPath), escapeshellarg($base));
        exec($cmd);
        return glob("$base-*.png") ?: [];
    }
}

// app/Services/WebClipService.php
namespace App\Services;

use Symfony\Component\DomCrawler\Crawler;

class WebClipService {
    public function extractMainText(string $html): string {
        $crawler = new Crawler($html);
        // タイトル＋見出し＋本文候補を抽出（簡易版）
        $parts = [];
        $title = $crawler->filter('title')->first();
        if ($title->count()) $parts[] = trim($title->text());
        $crawler->filter('h1,h2,h3,p,article')->each(function($n)
use (&$parts){
            $t = trim($n->text());
            if (mb_strlen($t) >= 10) $parts[] = $t;
        });
        return implode("\n\n", $parts);
    }
}
```

8) カード CRUD（Board/Detail/Update/Delete）

```
// app/Http/Controllers/CardController.php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Repositories\CardRepository;

class CardController extends Controller {
    public function board(CardRepository $repo) {
        return response()->json($repo->listSummaries());
    }
    public function show(string $id, CardRepository $repo) {
        $card = $repo->find($id);
        return $card ? response()->json($card) :
response()->json(['error'=>'not found'],404);
    }
    public function update(string $id, Request $req, CardRepository
$repo) {
        $validated = $req->validate([
            'summary.title' => 'nullable|string|max:200',
            'summary.company' => 'nullable|string|max:200',
            'detail.memo' => 'nullable|string|max:10000',
            'summary.tags' => 'nullable|array|max:50',
        ]);
        $card = $repo->find($id);
        if (!$card) return response()->json(['error'=>'not
found'],404);
        $merged = array_replace_recursive($card, $validated);
        $repo->upsert($id, $merged);
        return response()->json($merged);
    }
    public function destroy(string $id, CardRepository $repo) {
        $repo->delete($id);
        return response()->json(['deleted'=>true]);
    }
}
```

設定まわり

CORS

`config/cors.php` にフロントのオリジンを許可。

ファイル・制限

`POST /v1/documents` のバリデーションで **MIME** と **10MB** を厳格化済み。

キュー（非同期）

`.env`

```
QUEUE_CONNECTION=database
```

-

初期化

```
php artisan queue:table
php artisan migrate
php artisan queue:work --tries=1
```

-

本番は **Supervisor** で常駐
`supervisor.conf.sample`（要点）

```
[program:server-php-queue]
command=php /path/to/artisan queue:work --sleep=1 --tries=1
--timeout=120
autostart=true
autorestart=true
numprocs=1
redirect_stderr=true
stdout_logfile=/var/log/laravel-queue.log
```

-

永続化（**JSON**）

- 既存 Node 同様、「起動時ロード／変更時即保存」の動作を Repository が担保。
- 将来 DB（SQLite/PostgreSQL）移行時は Repository 差し替えのみ。

言語データ

- サーバに `tesseract-ocr`、`tesseract-ocr-jpn`、`tesseract-ocr-jpn-vert` をインストール。
- `OcrService` の `->lang('jpn','eng')` で指定。

進捗表示

- Node 版の Tesseract logger % と同様の UI を保ちつつ、Laravel 版は「PDF は ページ進捗」「画像/テキストは 段階進捗」で `%` を近似（上記ジョブで実装済み）。

PDF の canvas 依存の代替方針（PHP）

- サーバ側描画：`Imagick` か `pdftoppm` で画像化 → OCR（canvas 不要）
- **PDF** テキスト直抽出も併用可：`pdftotext` が入っていれば、まず `pdftotext` → 文字化け時だけ画像化 OCR にフォールバック（精度＆速度最適化）

フロント（React）の変更点

- エンドポイント **URL** を PHP 側に向けるだけで OK（パス互換）
- ポーリング間隔/最大回数は現行仕様のまま（500ms×60=30s）
- 進捗%は返しているが、UI は「スピナーのみ」で表示不要（互換）

セキュリティ & 運用メモ

- CORS、拡張子＆MIME、10MB 上限はサーバ側で強制
- 将来は：
    - 認証（Personal Access Token or Cookie＋CSRF）
    - レート制限（`throttle:60,1` など）
    - ログ構造化（`monolog`）
    - S3/Cloud Storage＋CDN（`Storage::disk('s3')` に切替）

初期セットアップ手順（最短）

```
# 1) プロジェクト
cd apps && composer create-project laravel/laravel server-php
cd server-php
composer require thiagoalessio/tesseract_ocr symfony/dom-crawler
symfony/css-selector

# 2) OS パッケージ
# (例: Ubuntu)
sudo apt-get install -y tesseract-ocr tesseract-ocr-jpn
tesseract-ocr-jpn-vert \
    imagemagick poppler-utils

# 3) キュー
php artisan queue:table
php artisan migrate

# 4) ストレージ準備
mkdir -p storage/app/data storage/app/uploads
echo "{}" > storage/app/data/cards.json
echo "{}" > storage/app/data/events.json
echo "{}" > storage/app/data/jobs.json
echo "{}" > storage/app/data/documents.json

# 5) 起動
php artisan serve
```

```
php artisan queue:work
```

既知の課題・改善（PHP 版）

- **OCR** 進捗の粒度：Tesseract の “正確な %” は得にくい → ページ進捗＋段階進捗のハイブリッドで代替
- **PDF** のレイアウト保持：`pdftotext -layout` → ダメな場合は画像 OCR 併用（改行維持は完全ではない）
- 縦書き混在：`jpn_vert` の適用やページ毎判定のロジック強化は今後の改善点
- **JSON** 同時書込：同時更新が多い場合は **DB** 移行（SQLite でも良い）を推奨

まとめ（移植のポイント）

- ルーティングとレスポンス形式は 既存 **API** と互換
- 非同期は Laravel Queue＋Supervisor で安定運用
- 永続化はまず **JSON**、将来 **DB** に差し替え容易
- PDF は サーバ側画像化で canvas 依存を解消
- React 側は **API** ベース **URL** の切替のみで動作