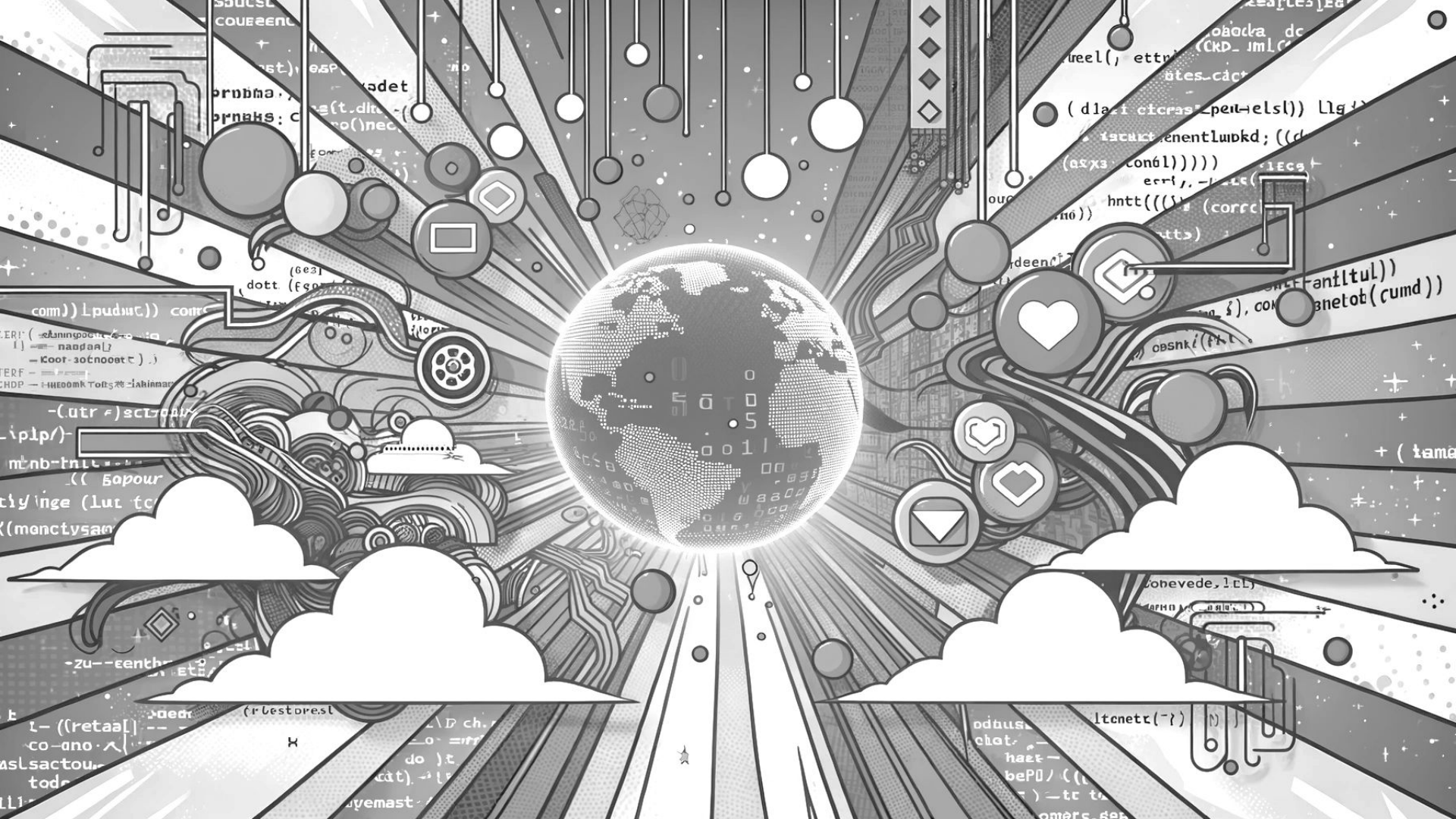
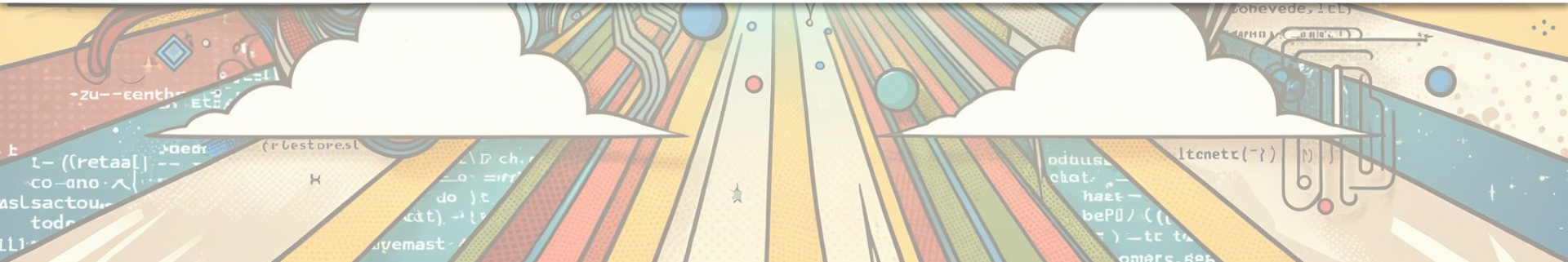


Jakub Ciszak

Starszy programista

Jak zadbać o czystą domenę?

(Nie rezygnując z udogodnień frameworka.)



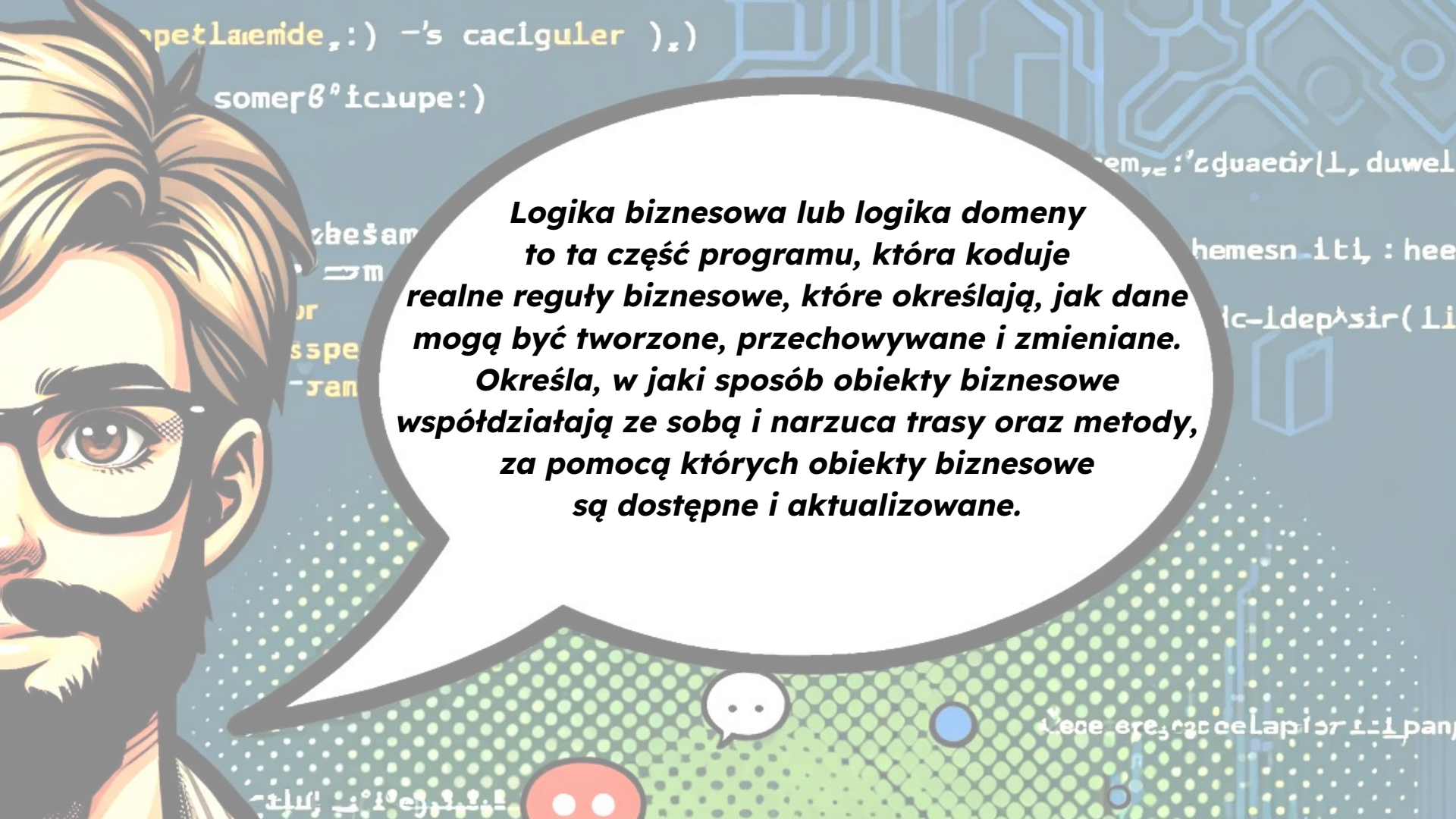
O czym będziemy mówić?

- Czym jest logika biznesowa?
- Dlaczego odseparowywać logikę?
- Request i DTO
- Tworzenie DTO, mapowanie requestu
- Command/Query
- Walidacja
- Encje
  - Programowanie w OOP
  - Mapowanie
  - Wartości
  - Repozytoria

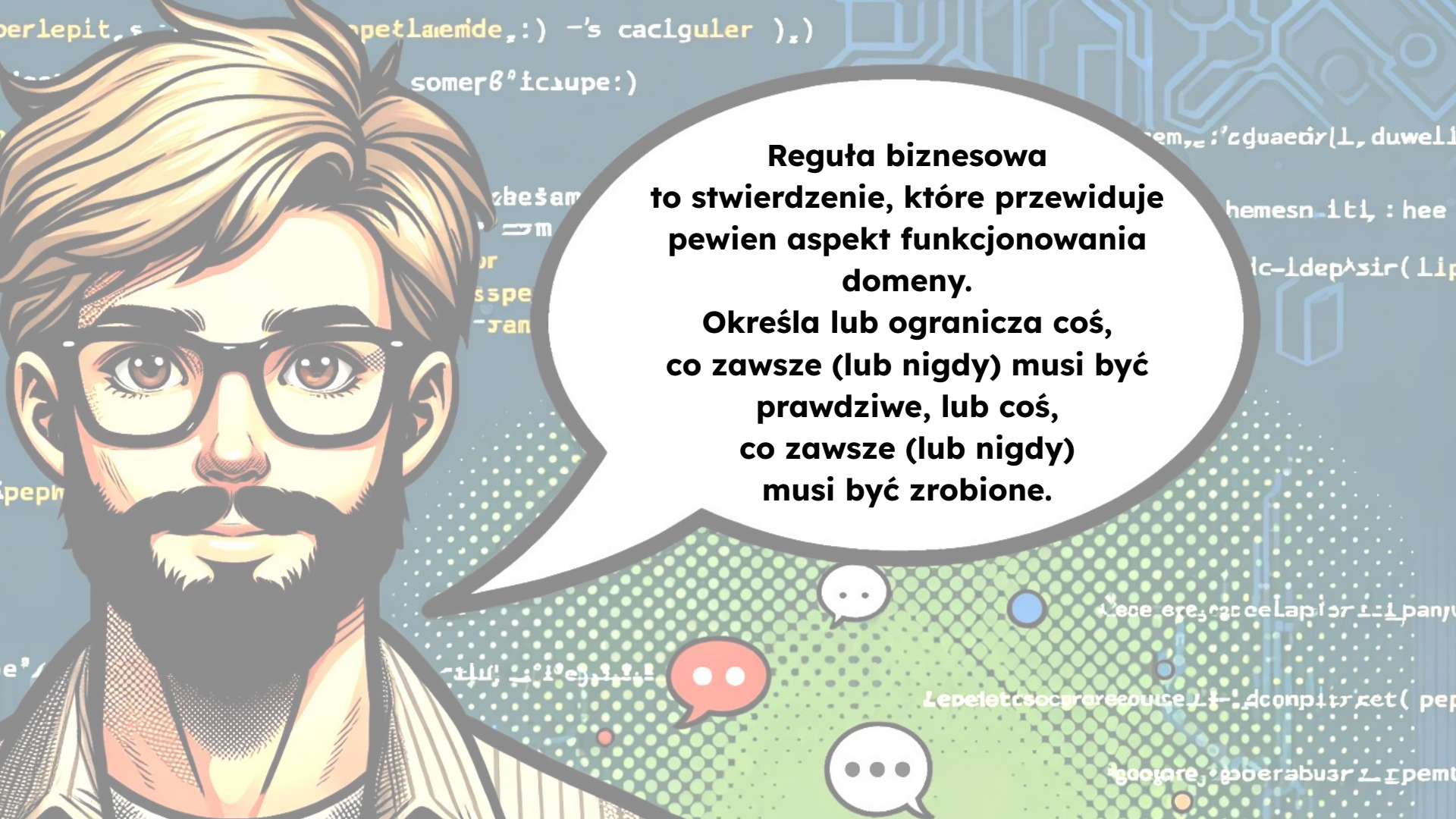
O czym będziemy mówić?

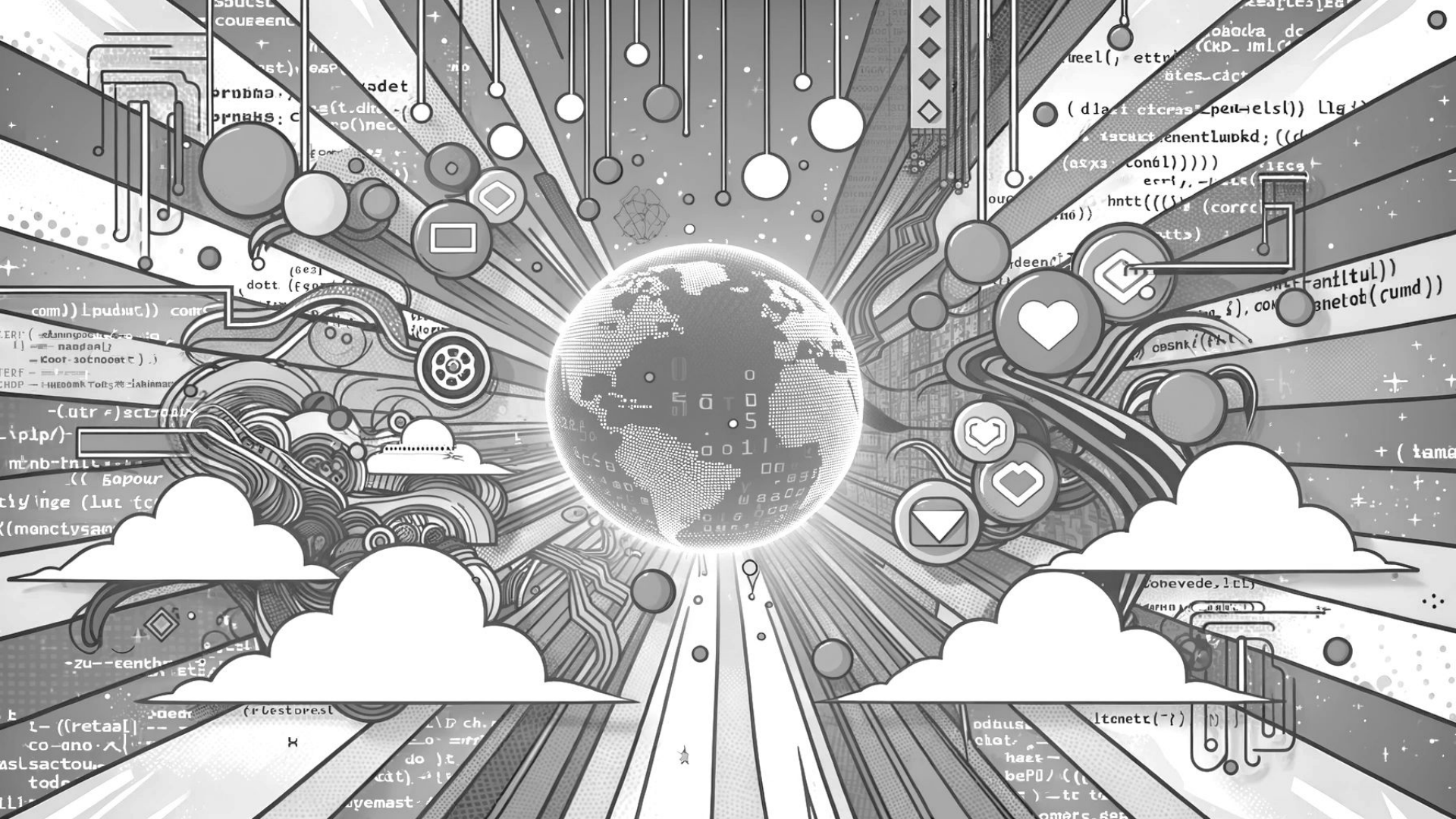
- Czym jest logika biznesowa?
- Dlaczego odseparowywać logikę?
- Request i DTO
- Tworzenie DTO, mapowanie requestu
- Command/Query
- Walidacja
- Encje
  - Programowanie w OOP
  - Mapowanie
  - Wartości
  - Repozytoria


LOGIKA


Logika biznesowa lub logika domeny
to ta część programu, która koduje
realne reguły biznesowe, które określają, jak dane
mogą być tworzone, przechowywane i zmieniane.
Określa, w jaki sposób obiekty biznesowe
współdziałają ze sobą i narzuca trasy oraz metody,
za pomocą których obiekty biznesowe
są dostępne i aktualizowane.

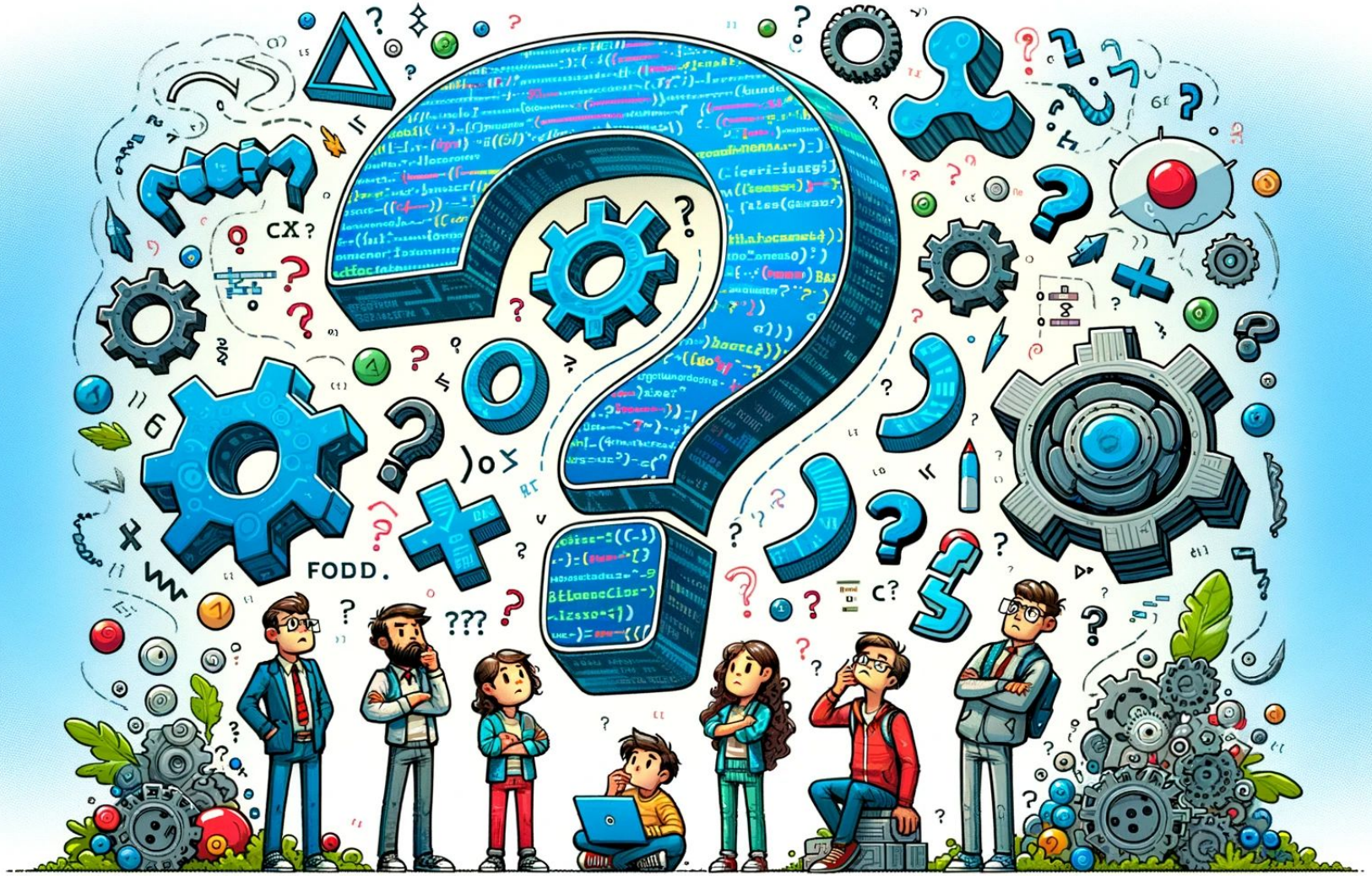

REGUŁA


Reguła biznesowa
to stwierdzenie, które przewiduje
pewien aspekt funkcjonowania
domeny.
Określa lub ogranicza coś,
co zawsze (lub nigdy) musi być
prawdziwe, lub coś,
co zawsze (lub nigdy)
musi być zrobione.



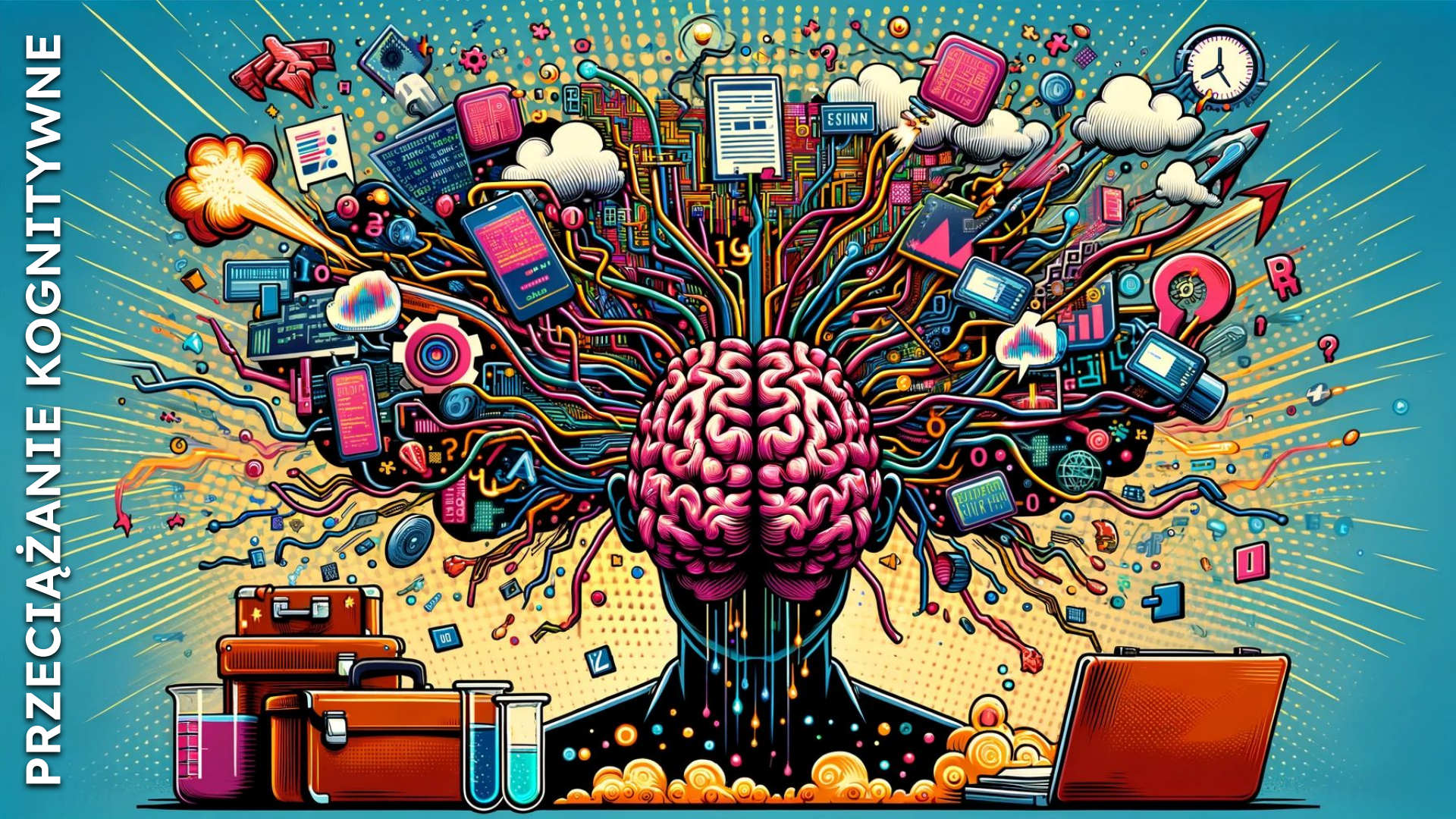
NARZĘDZIA

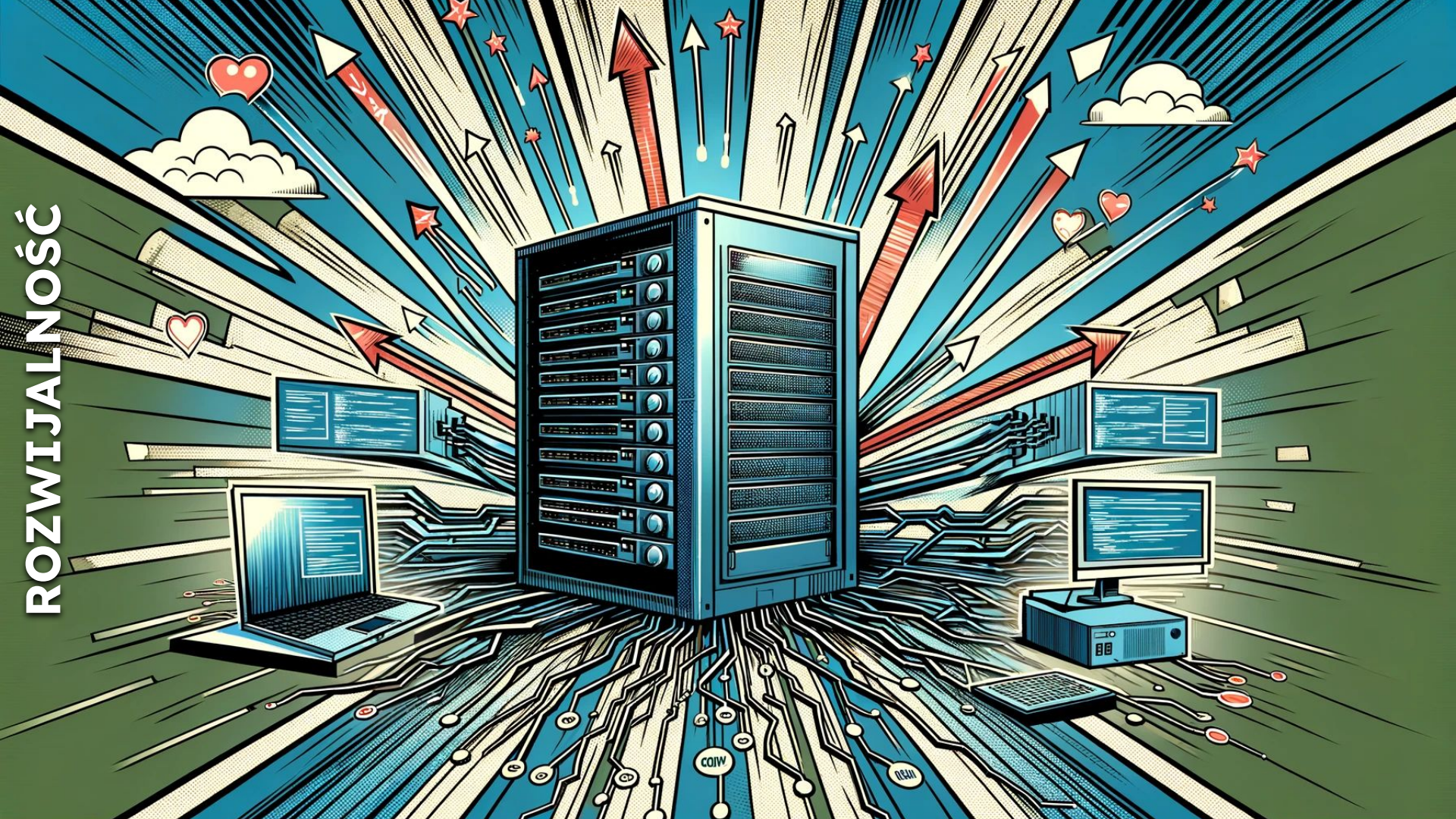
O co tyle hałasu?

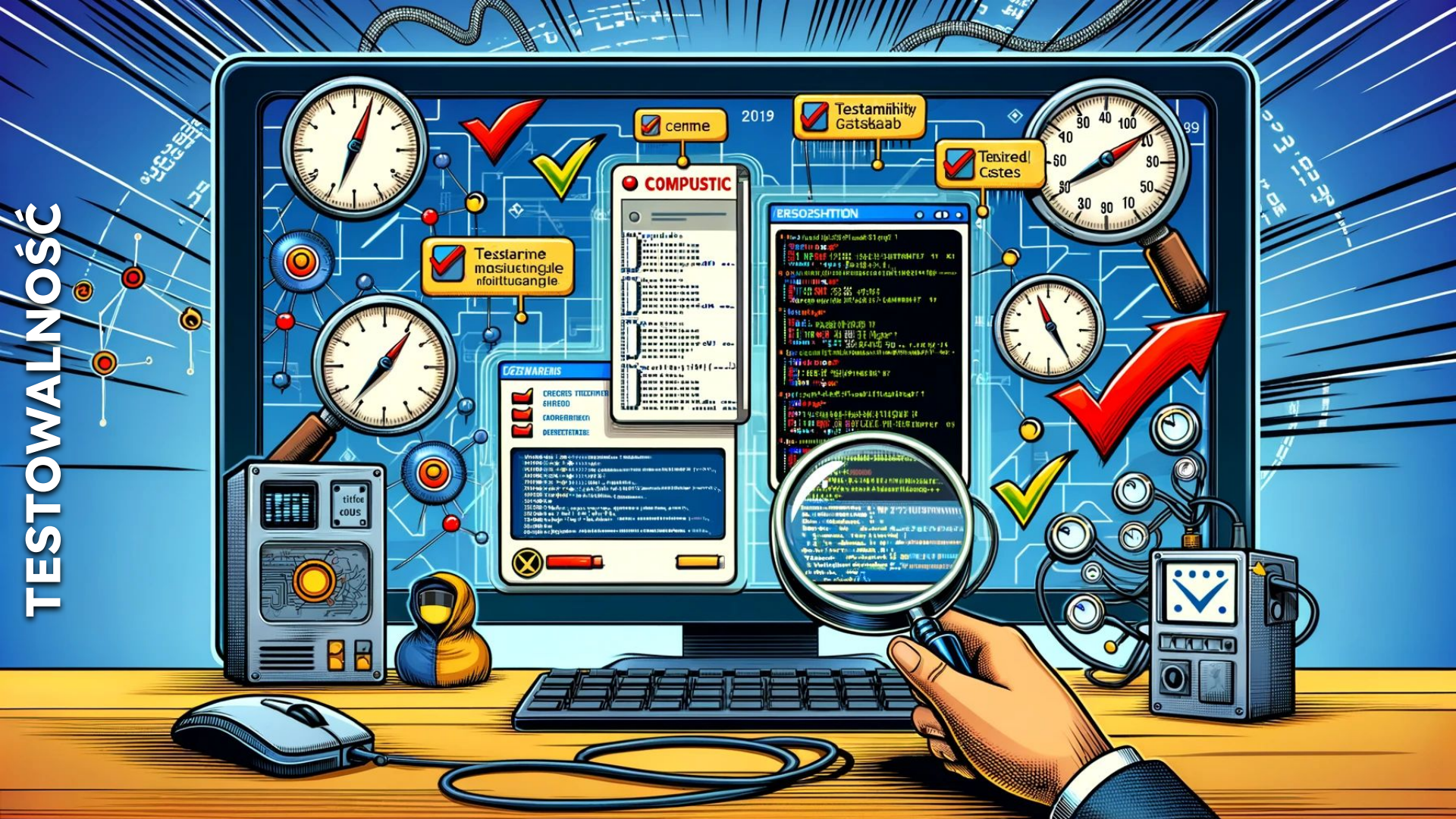


Po co odseparowywać logikę?

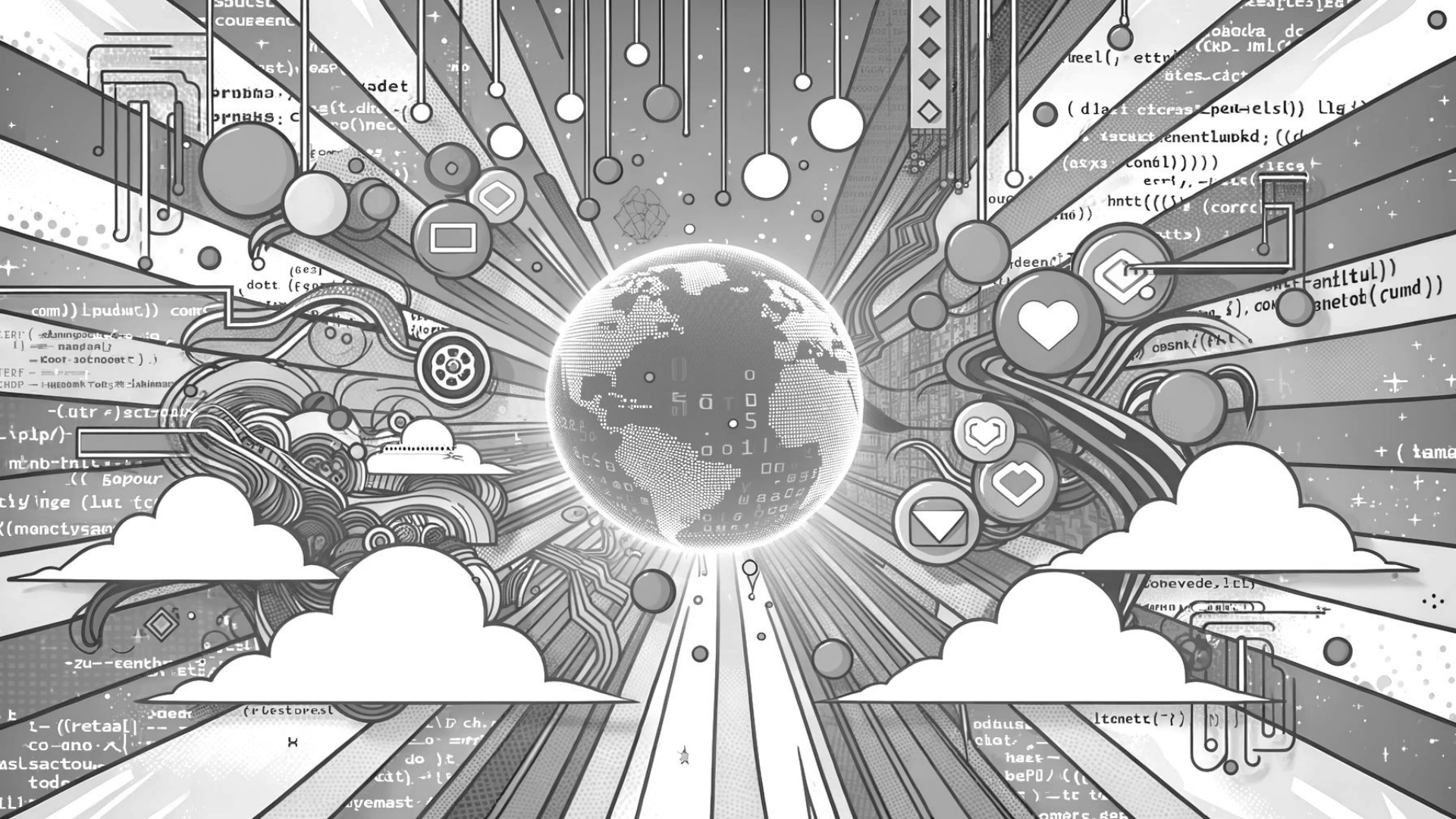
PRZECIĄŻANIE KOGNITYWNE

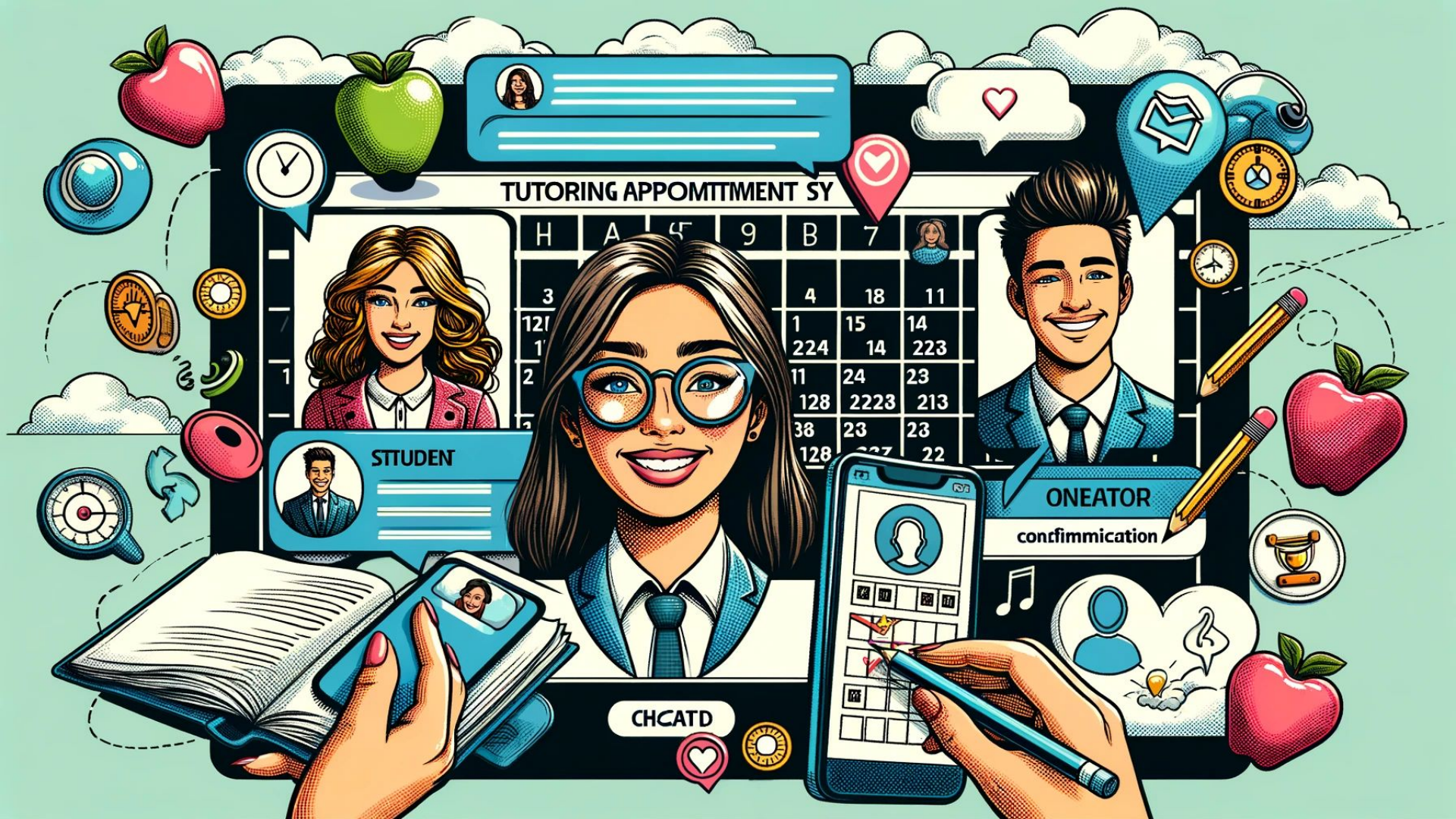


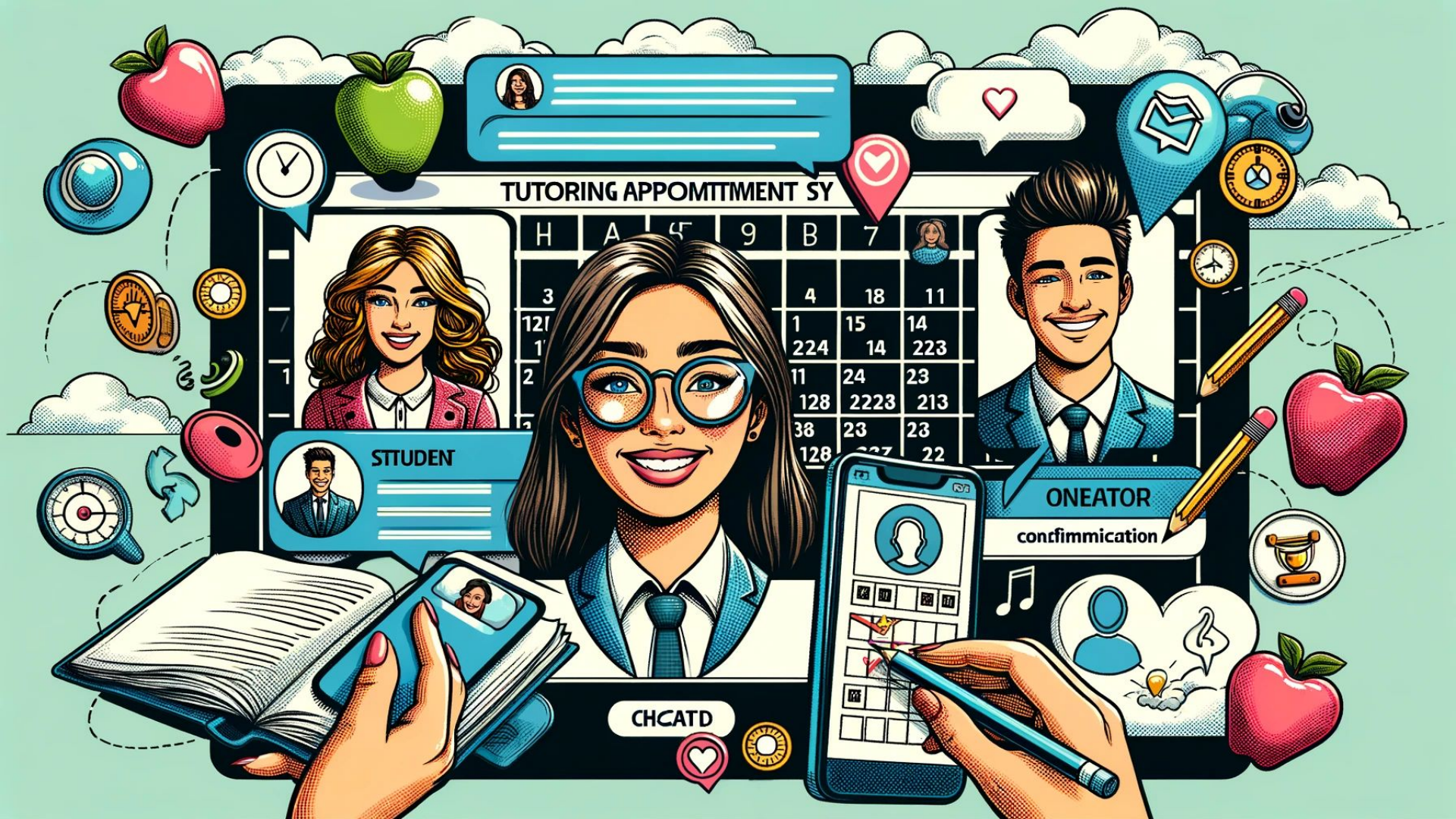

ROZWIJALNOŚĆ

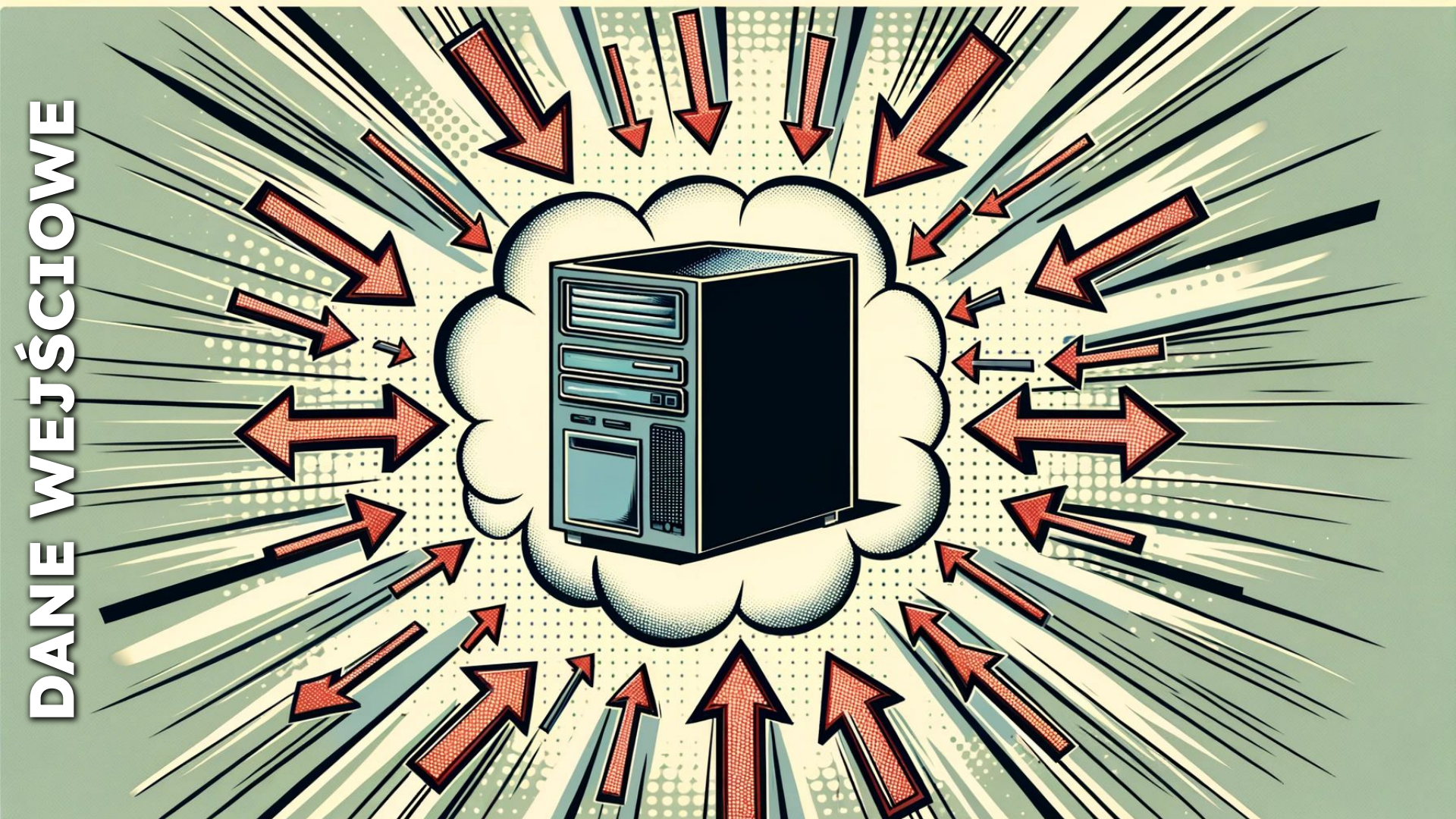

TESTOWALNOŚĆ


DOBRE SAMOPOCZUCIE




TUTORING APPOMTTMENT SY
STTUDEN
ONEATOR
confimmication
CHCATD

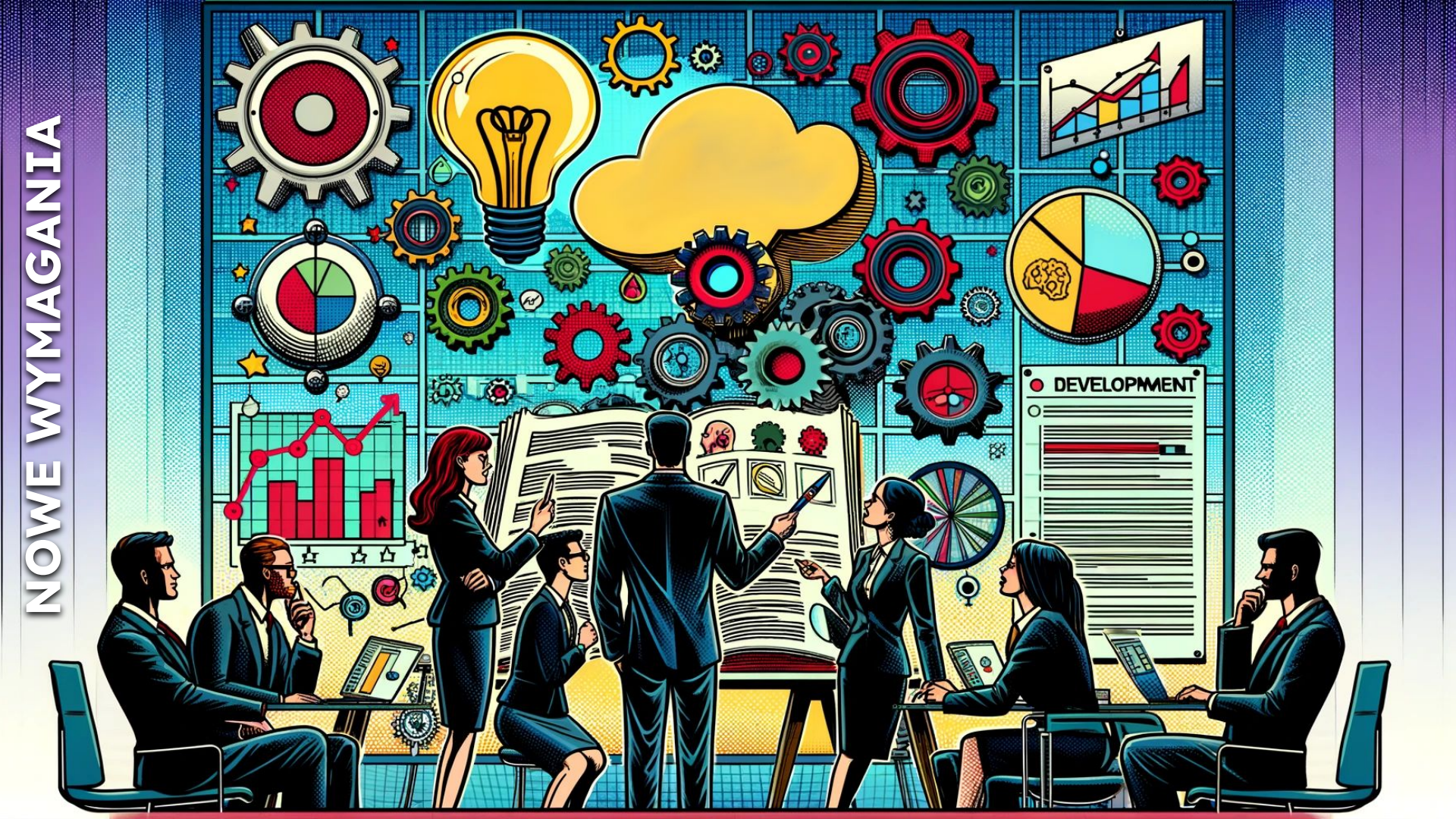

TUTORING APPOMTTMENT SY
STTUDEN
ONEATOR
confimmication
CHCATD


DANE WEJŚCIOWE

```php
#[Route('/schedule', name: 'schedule')]
public function index(Request $request, LessonRepository $lessonRepository): Response
{
    $data = $request->toArray();
    $criteria = Criteria::create();
    if (empty($data['userId'])) {
        $this->addFlash( type: 'danger', message: 'User id is required');
}
    $criteria->andWhere(Criteria::expr()->eq( 'userId', $data['userId']));

    $subject = $data['subject'] ?? null;
    if ($subject && strlen($subject) < 3) {
        $this->addFlash( type: 'danger', message: 'Subject must be at least 3 characters long');
    }
    $criteria->andWhere(Criteria::expr()->eq('subject', $subject));

    $dateFrom = new \DateTime($data['dateFrom']) ?? null;
    $dateTo = new \DateTime($data['dateTo']) ?? null;
    if ($dateFrom) {
        $criteria->andWhere(Criteria::expr()->gte( 'date', $dateFrom));
    }
    if ($dateTo) {
        $criteria->andWhere(Criteria::expr()->lte( 'date', $dateTo));
    }
    if ($dateFrom && $dateTo) {
        if ($dateFrom > $dateTo) {
            $this->addFlash( type: 'danger', message: 'Date from must be before date to');
        }
    }

    $lessons = $lessonRepository->findBy($criteria);
    return $this->render( view: 'schedule/index.html.twig', [
        'lessons' => $lessons,
    ]);
}
```

```php
#[Route('/api/schedule', name: 'api_schedule')]
public function index(
    Request $request,
    LessonListService $lessonListService
): Response {
    $filters = [
        'subject' => $request->get( key: 'subject'),
        'dateFrom' => $request->get( key: 'dateFrom'),
        'dateTo' => $request->get( key: 'dateTo'),
    ];
    $result = $lessonListService->getLessonsByFilters($filters);
    if ($result->hasErrors()) {
        foreach ($result->getErrors() as $error) {
            $this->addFlash( type: 'danger', $error->getMessage());
        }
    }

    return $this->render(
        view: 'schedule/index.html.twig',
        [
            'lessons' => $result->getLessons(),
        ]
    );
}
```

```php
public function getLessonsByFilters(array $filters): LessonListResponse
{
    $errors = $this->validateFilters($filters);
    $response = new LessonListResponse();
    if (!empty($errors)) {
        $response->setErrors($errors);
        return $response;
    }
    $criteria = Criteria::create();

    $this->applyUserIdFilter($filters, $criteria);
    $this->applySubjectFilter($filters, $criteria);
    $this->applyDateFilter($filters, $criteria);

    $lessons = $lessonRepository->findBy($criteria);
    $response->setLessons($lessons);
    return $response;
}
```

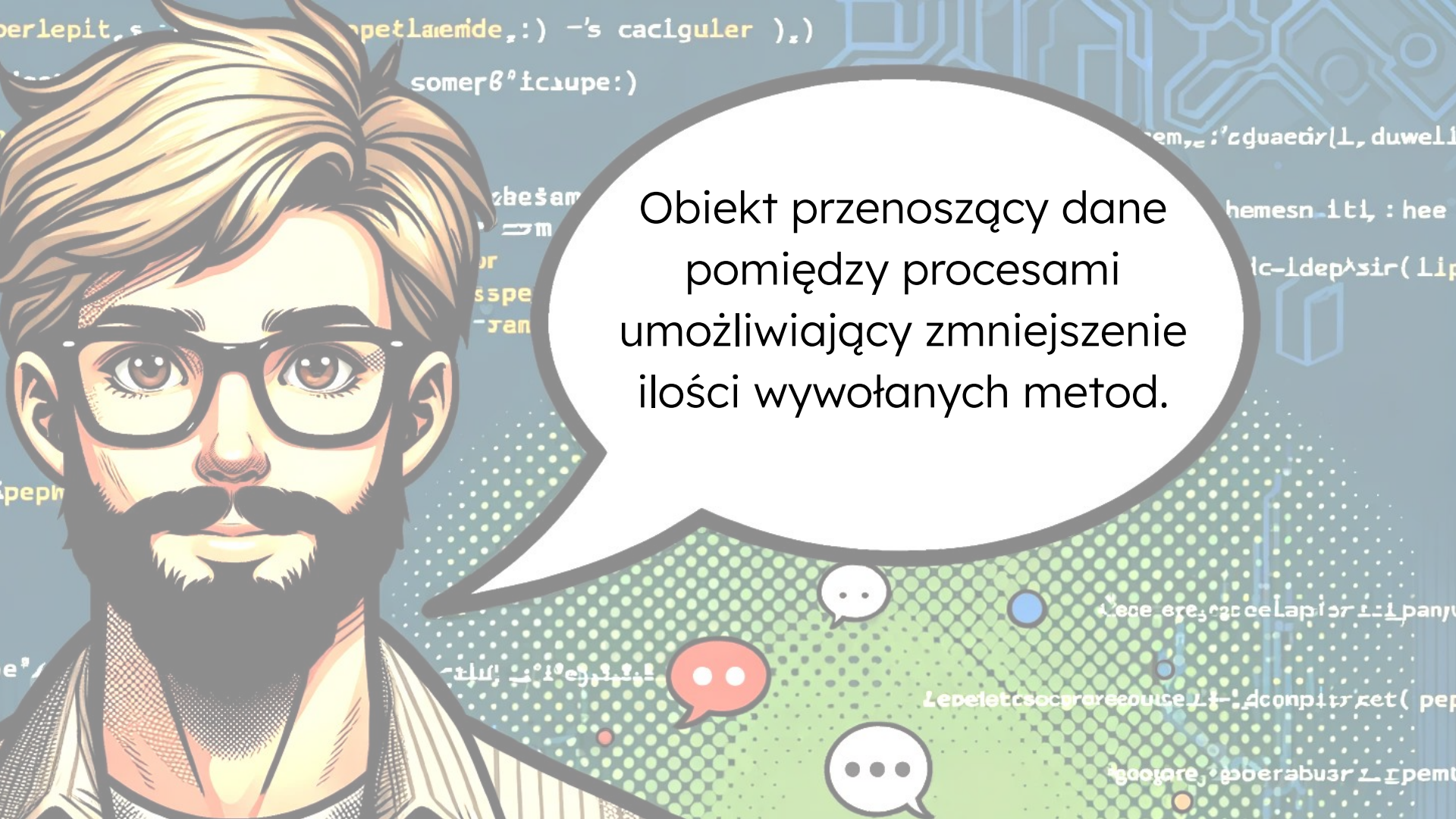

NOWE WYMAGANIA
DEVELOPMENT

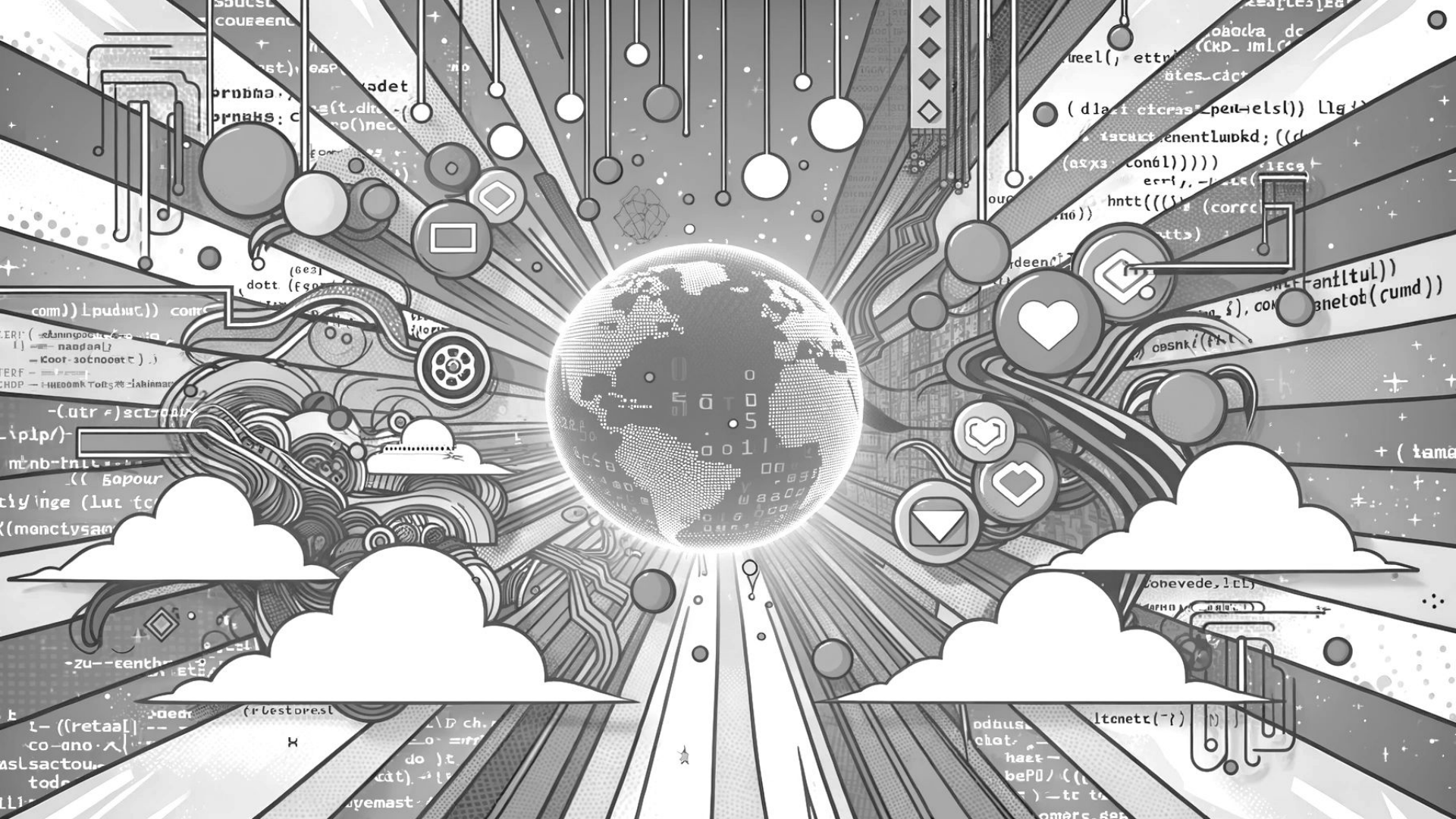
```php
#[Route('/lessons', name: 'lessons_list')]
public function getLessonsList(Request $request, LessonListService $lessonListService): Response
{
    $filters = [
        'subject' => $request->get( key: 'subject'),
        'dateFrom' => new \DateTimeImmutable($request->get( key: 'dateFrom')),
        'dateTo' => new \DateTimeImmutable($request->get( key: 'dateTo'))
    ];

    $lessonListService->getLessonsByFilters($filters);

    //...
}
```

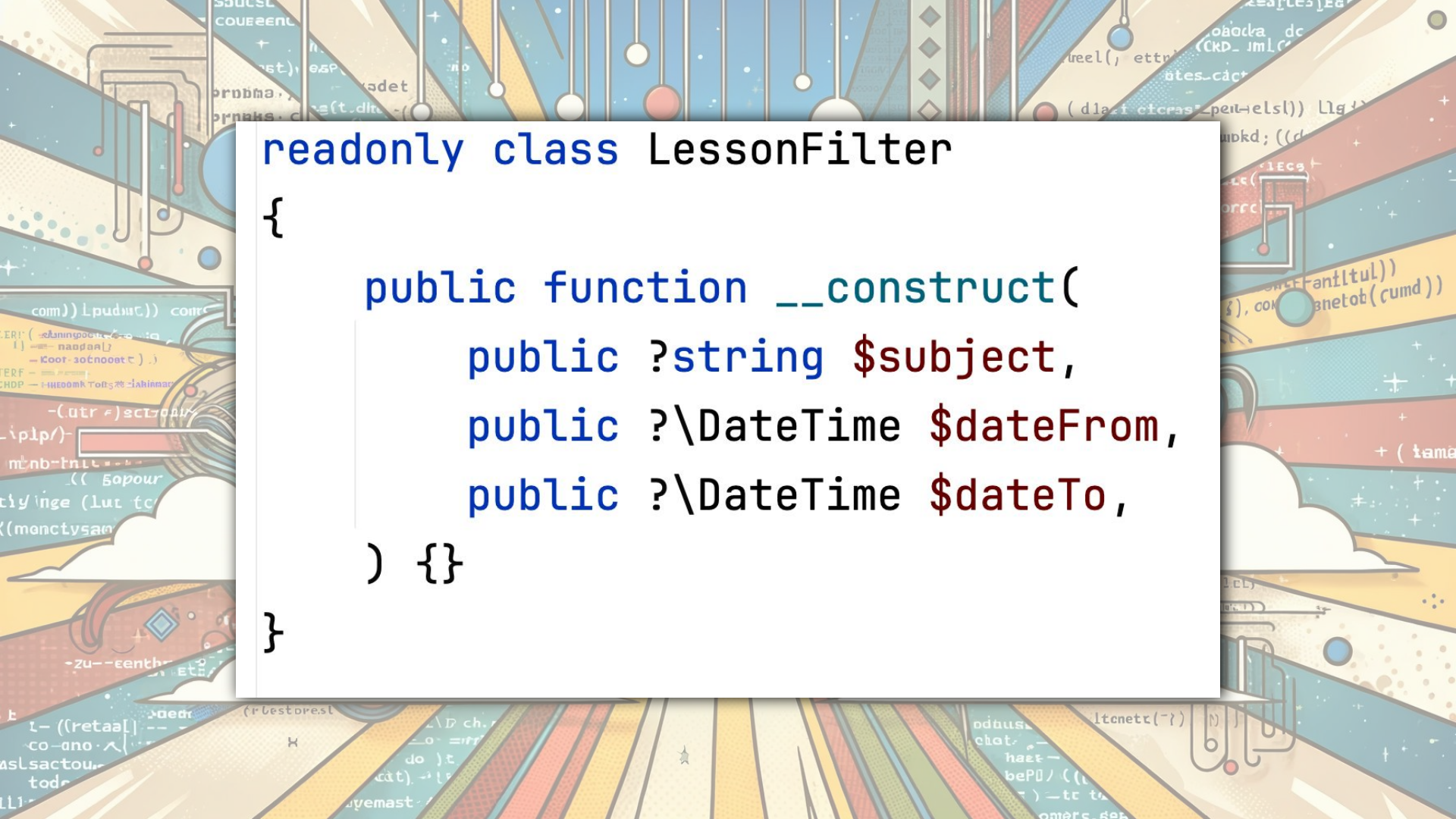
```php
#[Route('/api/schedule', name: 'api_schedule')]
public function index(
    Request $request,
    LessonListService $lessonListService
): Response {
    $filters = [
        'subject' => $request->get( key: 'subject'),
        'dateFrom' => $request->get( key: 'dateFrom'),
        'dateTo' => $request->get( key: 'dateTo'),
    ];

ervice->getLessonsByFilters($filters);
()) {
```

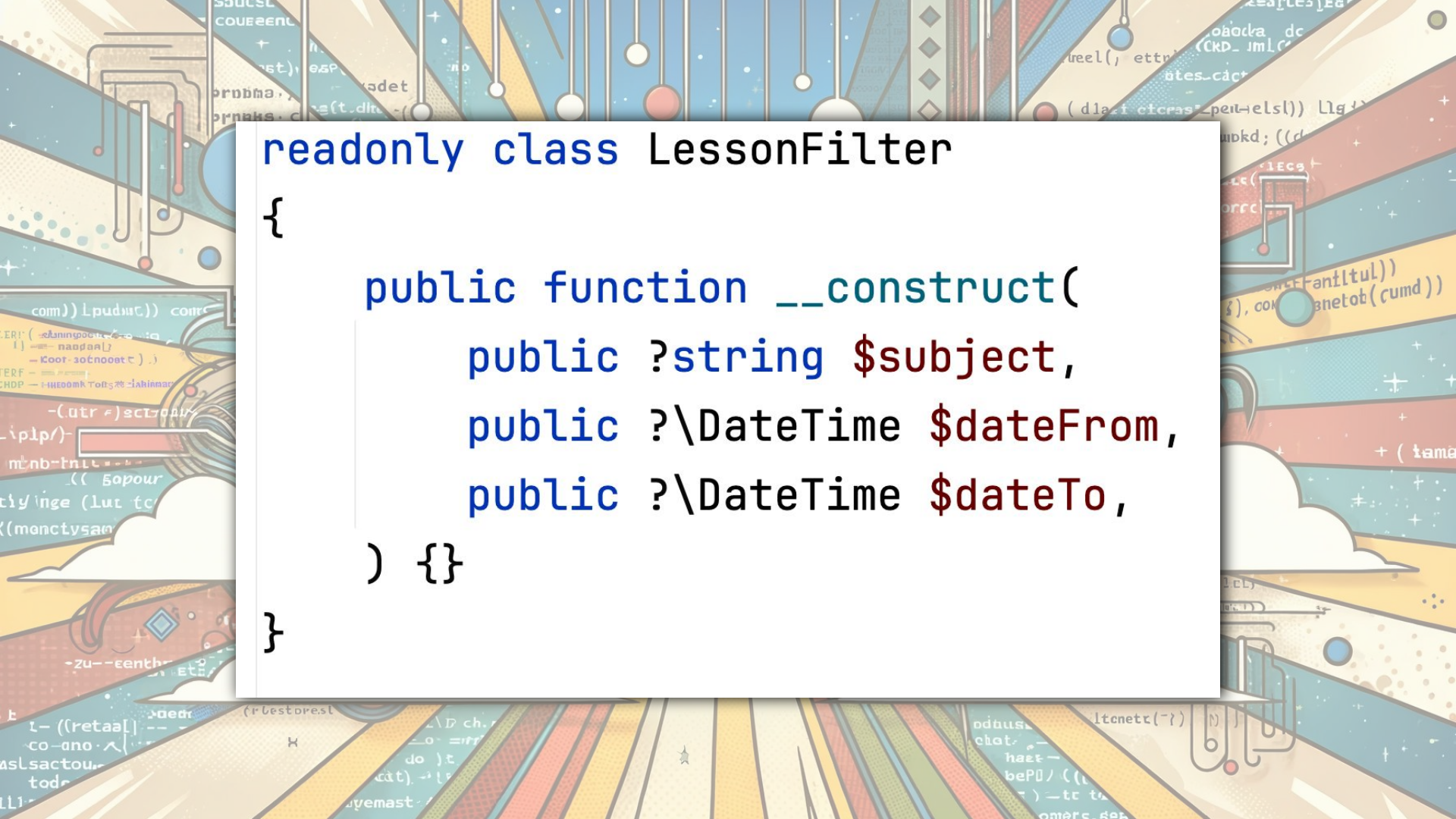
```php
protected function execute(InputInterface $input, OutputInterface $output): int
{
    $notificationConfigs = $this->notificationConfigRepository->findForActiveUsers();
    foreach ($notificationConfigs as $notificationConfig) {
        $filter = $notificationConfig->getFilterArray();
        $lessonsResponse = $this->lessonListService->getLessonsByFilters($filter);
        foreach ($lessonsResponse->getLessons() as $lesson) {
            $this->notificationService->sendNotification($notificationConfig, $lesson);
        }
    }
    return 0;
}
```


DTO


Obiekt przenoszący dane pomiędzy procesami umożliwiający zmniejszenie ilości wywołanych metod.




readonly class LessonFilter
{
public function __construct(
public ?string $subject,
public ?\DateTime $dateFrom,
public ?\DateTime $dateTo,
) {}
}


readonly class LessonFilter
{
public function __construct(
public ?string $subject,
public ?\DateTime $dateFrom,
public ?\DateTime $dateTo,
) {}
}

Po co DTO?

- Podział odpowiedzialności
- Kontrola nad danymi wejściowymi
- Klarowność
- Testowalność
- Reużywalność

Po co DTO?

- Podział odpowiedzialności
- Kontrola nad danymi wejściowymi
- Klarowność
- Testowalność
- Reużywalność

```php
#[Route('/lessons', name: 'lessons_list')]
public function getLessonsList(Request $request, LessonListService $lessonListService): Response
{
    $filters = new LessonFilter(
        $request->get( key: 'subject'),
        new \DateTimeImmutable($request->get( key: 'dateFrom')),
        new \DateTimeImmutable($request->get( key: 'dateTo'))
    );

    $lessonListService->getLessonsByFilters($filters);

    //...
}
```

- Skupienie na obsłudze żądań i przekazywaniu danych do warstwy aplikacji

- Scentralizowanie tworzenia obiektów DTO ułatwia zarządzanie nimi w całej aplikacji.

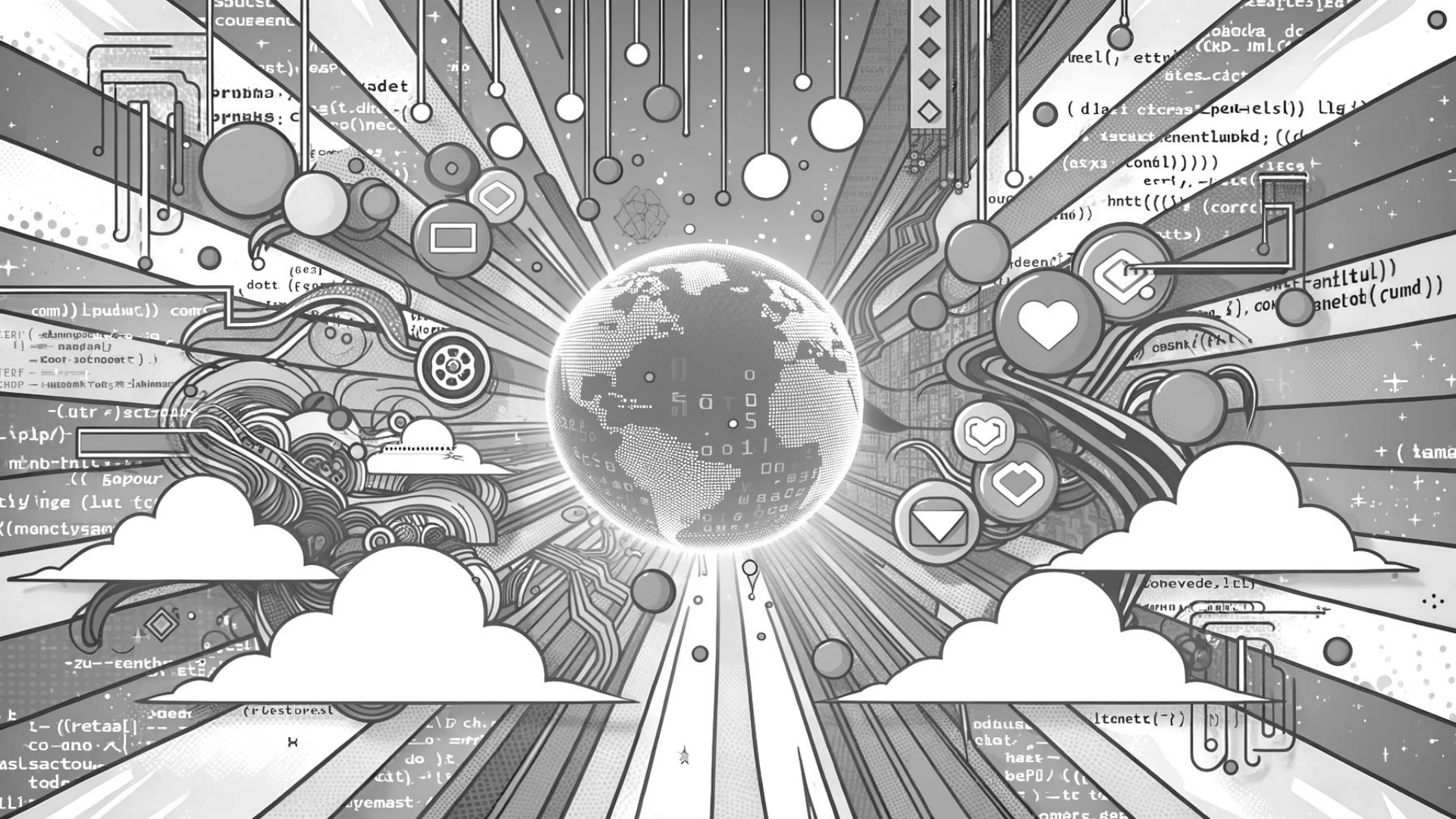

FACTORY

```php
class LessonFilterFactory
{
    public function create(array $data): LessonFilter
    {
        return new LessonFilter(
            $data['subject'],
            new \DateTimeImmutable($data['dateFrom']),
            new \DateTimeImmutable($data['dateTo'])
        );
    }
}
```

```php
protected function execute(InputInterface $input, OutputInterface $output): int
{
    $notificationConfigs = $this->notificationConfigRepository->findForActiveUsers();
    foreach ($notificationConfigs as $notificationConfig) {
        $filter = $this->lessonFilterFactory->create(
            $notificationConfig->getFilterArray()
        );
        $lessonsResponse = $this->lessonListService->
        foreach ($lessonsResponse->getLessons() as $
            $this->notificationService->sendNotifica
        }
    }
    return 0;
}
```

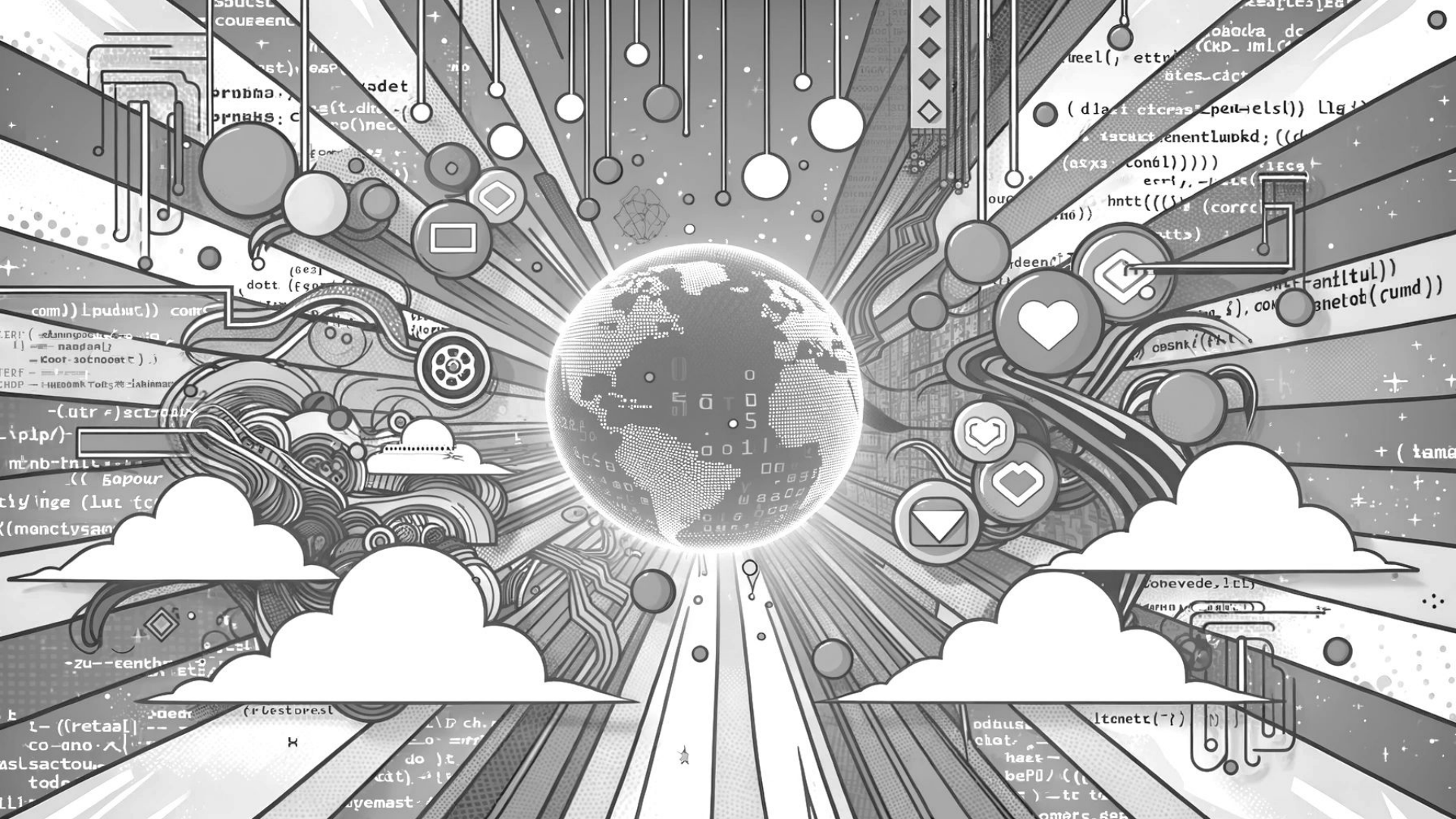
```php
#[Route('/lessons', name: 'lessons_list')]
public function getLessonsList(
    Request $request,
    LessonListService $lessonListService,
    LessonFilterFactory $lessonFilterFactory
): Response {
    $filters = $lessonFilterFactory->create($request->toArray());
    $lessonListService->getLessonsByFilters($filters);
```

```php
#[Route('/notification/configure', name: 'notification_configure')]
public function notificationConfigure(
    Request $request,
    NotificationService $notificationService,
    LessonFilterFactory $lessonFilterFactory
): Response {
    $filters = $lessonFilterFactory->create($request->toArray());
    $notificationService->configure($filters);
```

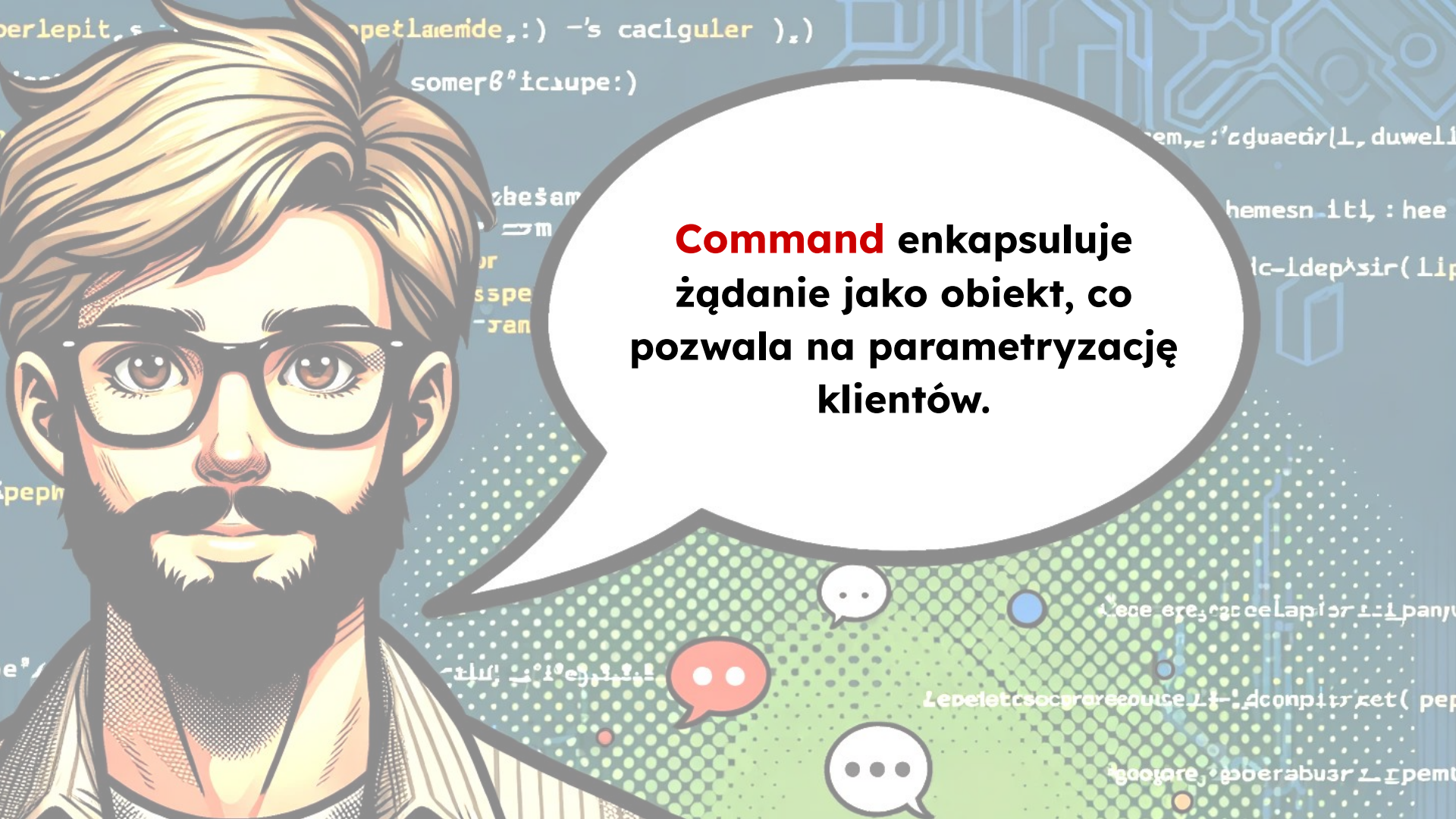


```php
#[Route('/lessons', name: 'lessons_list')]
public function getLessonsList(
    #[MapRequestPayload] LessonFilter $lessonFilter,
    LessonListService $lessonListService,
): Response {
    $lessonListService->getLessonsByFilters($lessonFilter);

    //...
}
```

```php
#[Route('/lessons', name: 'lessons_list')]
public function getLessonsList(
    #[MapRequestPayload] LessonFilter $lessonFilter,
    LessonListService $lessonListService,
): Response {
    $lessonListService->getLessonsByFilters($lessonFilter);

    //...
}
```




XXX enkapsuluje żądanie jako obiekt, co pozwala na parametryzację klientów.

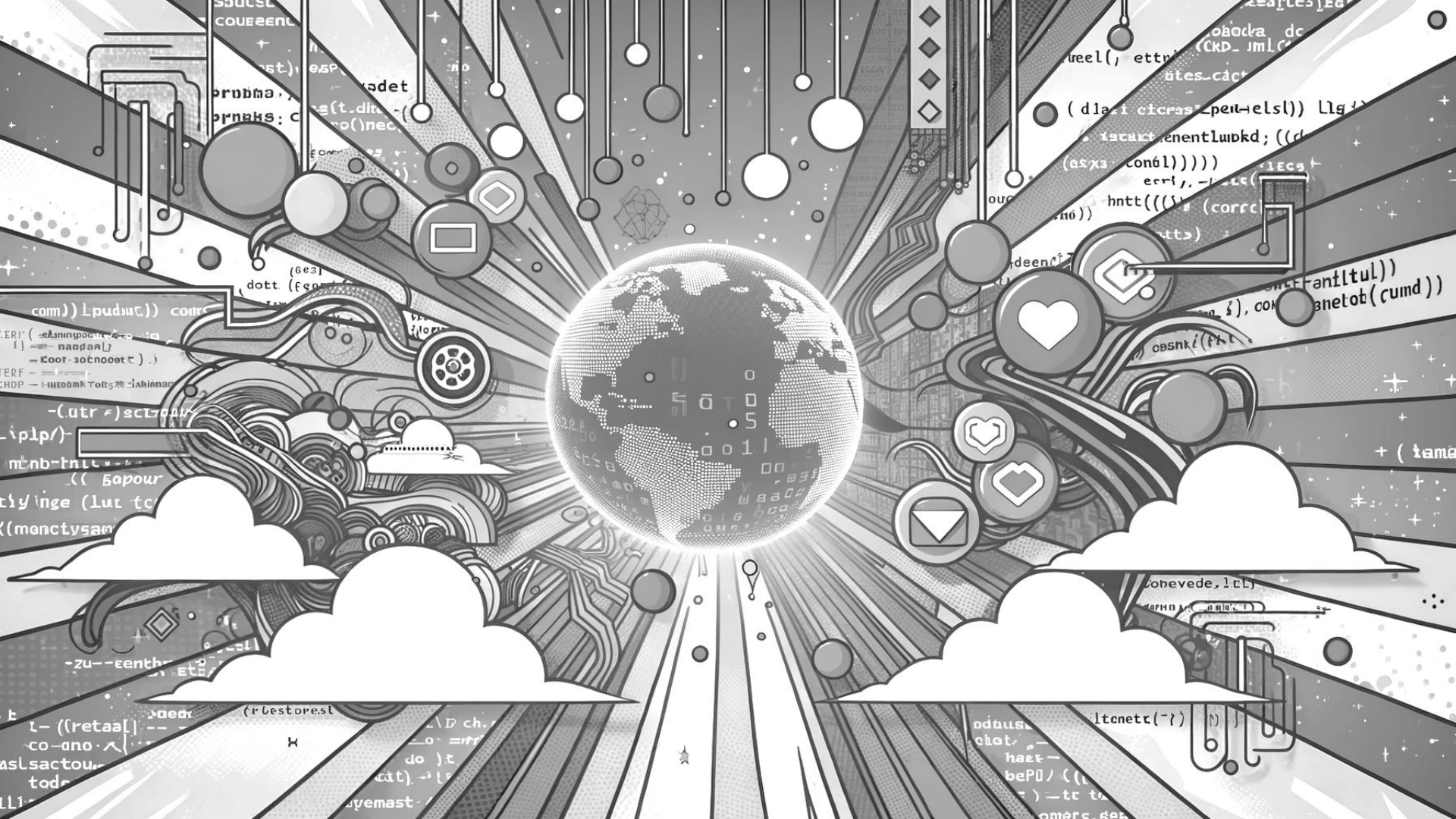

Command enkapsuluje żądanie jako obiekt, co pozwala na parametryzację klientów.


COMAND

```php
#[Route('/lesson/new', name: 'lesson_new')]
public function newLesson(
    Request $request,
    NewLessonCommandFactory $newLessonCommandFactory
): Response {
    $newLessonCommand = $newLessonCommandFactory->create($request);
    $this->messageBus->dispatch($newLessonCommand);

    //...
}
```

```php
readonly class CreateLessonCommandHandler implements CommandHandler
{
    public function __construct(
        private CreateLessonService $createLessonService
    ) {
    }

    public function __invoke(CreateLessonCommand $command): void
    {
        // ...   You, Moments ago • Uncommitted changes

        $this->createLessonService->execute($command);
    }
}
```

```
_instanceof:
    App\App\Command\Handler\CommandHandler:
        tags:
            - { name: messenger.message_handler }
```

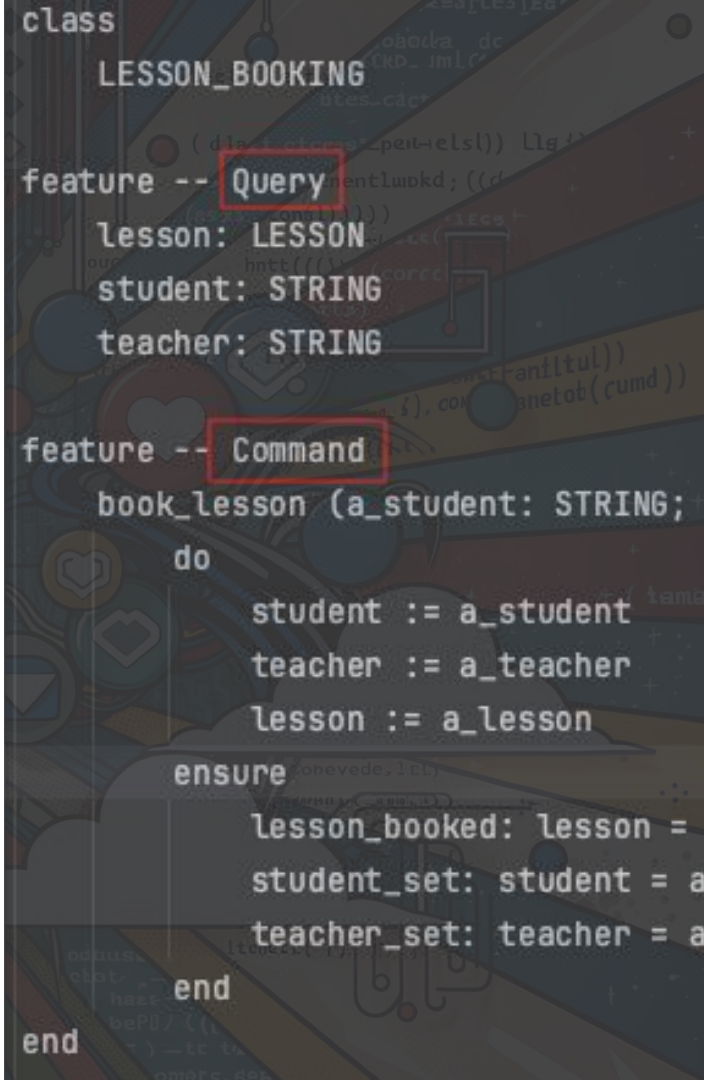



CQS

CQS

Command Query Separation

- **Oddzielenie operacji zmieniających stan systemu**
  od zapytań zwracających dane o nim.
- **Nie CQRS**
- **Nie ma wymogu stosowania oddzielnego modelu odczytu.**



```
class
    LESSON_BOOKING

feature -- Query
    lesson: LESSON
    student: STRING
    teacher: STRING

feature -- Command
    book_lesson (a_student: STRING;
        do
            student := a_student
            teacher := a_teacher
            lesson := a_lesson
        ensure
            lesson_booked: lesson =
            student_set: student = a
            teacher_set: teacher = a
        end
end
```


QUERY

```php
readonly class GetLessonDataQuery
{
    public function __construct(public string $lessonId)
    {
    }
}


class GetLessonDataQueryHandler implements QueryHandler
{
    //...
}
```

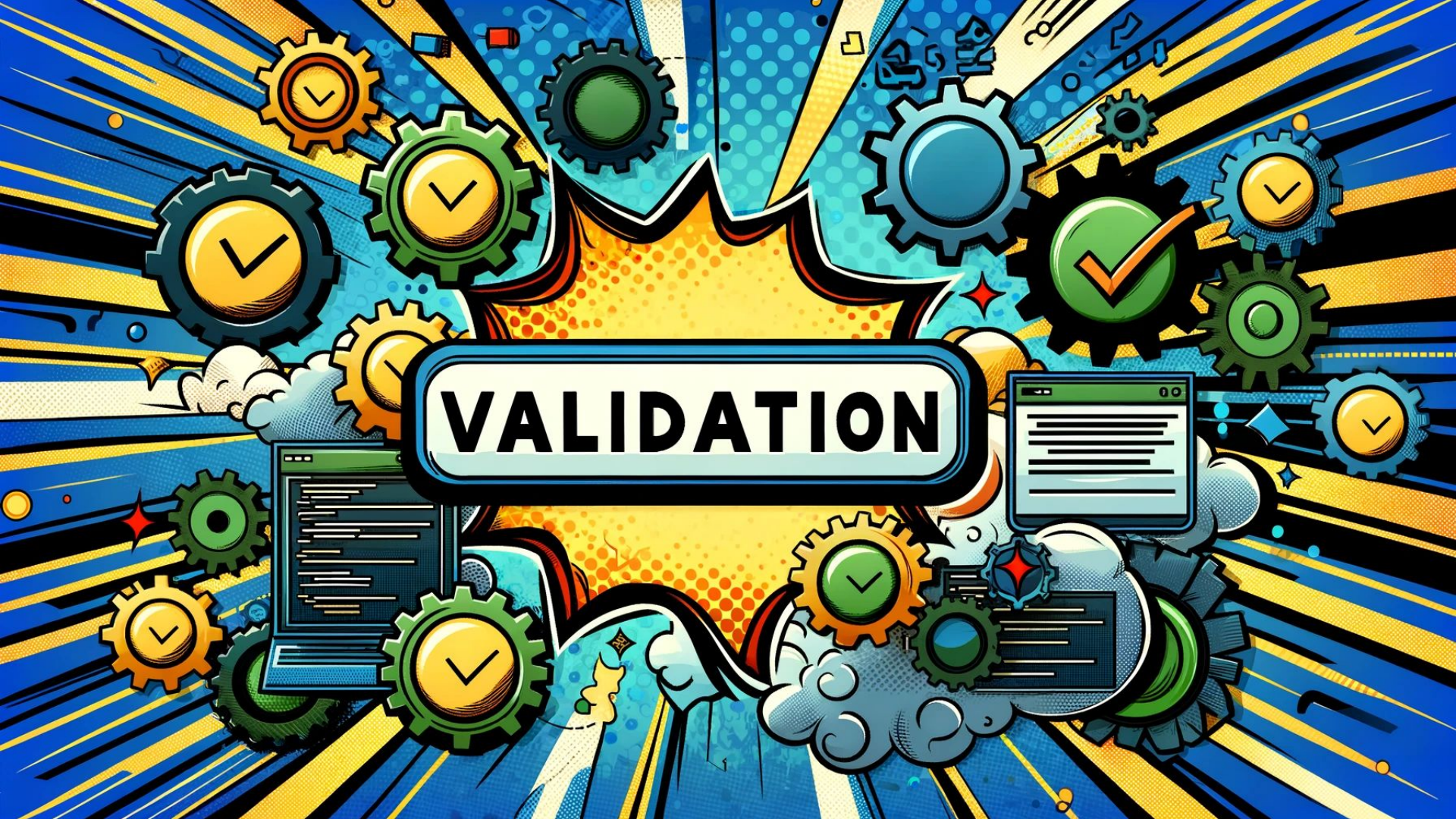
```
_instanceof:
    App\App\Command\Handler\CommandHandler:
        tags:
            - { name: messenger.message_handler }
    App\App\Query\Handler\QueryHandler:
        tags:
            - { name: messenger.message_handler }
```

```php
#[Route('/api/lesson/{lessonId}', name: 'lesson_data')]
public function getLessonData(string $lessonId): Response
{
    $envelope = $this->messageBus->dispatch(new GetLessonDataQuery($lessonId));
    $handledStamps = $envelope->all( stampFqcn: HandledStamp::class);
    if (!$handledStamps) {
        throw new LogicException(sprintf(
            format: 'Message of type "%s" was no handled',
            ...values: GetLessonDataQuery::class
        ));
    }
    /** @var LessonDataQueryResult $result */
    $result = $handledStamps[0]->getResult();
    return new JsonResponse($result->getLesson());
}
```


QUERY BUS
QUERY BUS

```php
class QueryBus
{
    use HandleTrait;

    public function __construct(MessageBusInterface $queryBus)
    {
        $this->messageBus = $queryBus;
    }

    public function handleQuery(QueryInterface $query): QueryResultInterface
    {
        return $this->handle($query);
    }
}
```

```php
class QueryBus
{
    use HandleTrait;

    public function __construct(MessageBusInterface $queryBus)
    {
        $this->messageBus = $queryBus;
    }

    public function handleQuery(QueryInterface $query): QueryResultInterface
    {
        return $this->handle($query);
    }
}
```


VALIDATION


ASSERT

```php
readonly class LessonFilter
{
    public function __construct(
        [redacted]
        [redacted]
        #[Assert\GreaterThan(propertyPath: 'dateFrom')]
        public ?\DateTime $dateTo,
    ) {}
}
```

Walidacja automatyczna przy mapowaniu

```php
#[Route('/lessons', name: 'lessons_list')]
public function getLessonsList(
    #[MapRequestPayload] LessonFilter $lessonFilter,
    LessonListService $lessonListService,
): Response {
    $lessonListService->getLessonsByFilters($lessonFilter);

    //...
}
```

Walidacja automatyczna przy mapowaniu

- Automatyzacja
- Czystość Kontrolerów
- Skupienie na Domenie

Walidacja automatyczna przy mapowaniu

- Automatyzacja
- Czystość Kontrolerów
- Skupienie na Domenie

- Magia Frameworka
- Event subscriber do błędów
- Słaba reużywalność

Walidator wstrzyknięty do fabryki

```php
#[Route('/lesson/new', name: 'lesson_new')]
public function newLesson(
    Request $request,
    NewLessonCommandFactory $newLessonCommandFactory,
    Validator $validator
): Response {
    $newLessonCommand = $newLessonCommandFactory->create($request, $validator);
    if (!$validator->isSuccess()) {
        foreach ($validator->getViolations() as $violation) {
            $this->addFlash( type: 'danger', $violation->getMessage());
        }
    }
```

Walidator wstrzyknięty do fabryki

- Enkapsulacja
- Separacja Odpowiedzialności
- Mniejsze ryzyko nieprawidłowych obiektów

Walidator wstrzyknięty do fabryki

- Enkapsulacja
- Reużywalność
- Separacja Odpowiedzialności

- Złożoność
- Dodatkowa praca

Walidacja przez middleware

```yaml
framework:
    messenger:
        default_bus: default.bus
        buses:
            default.bus:
                middleware:
                    - validation
```

Walidacja przez middleware

```php
#[Route('/lesson/new', name: 'lesson_new')]
public function newLesson(
    Request $request,
    NewLessonCommandFactory $newLessonCommandFactory
): Response {
    $newLessonCommand = $newLessonCommandFactory->create($request);

    try {
        $this->messageBus->dispatch($newLessonCommand);
    } catch (ValidationFailedException $exception) {
        foreach ($exception->getViolations() as $violation) {
            $this->addFlash( type: 'danger', $violation->getMessage());
        }
    }

    //...
}
```

Walidacja przez middleware

- Automatyzacja
- Czystość Kodu
- Centralizacja Walidacji

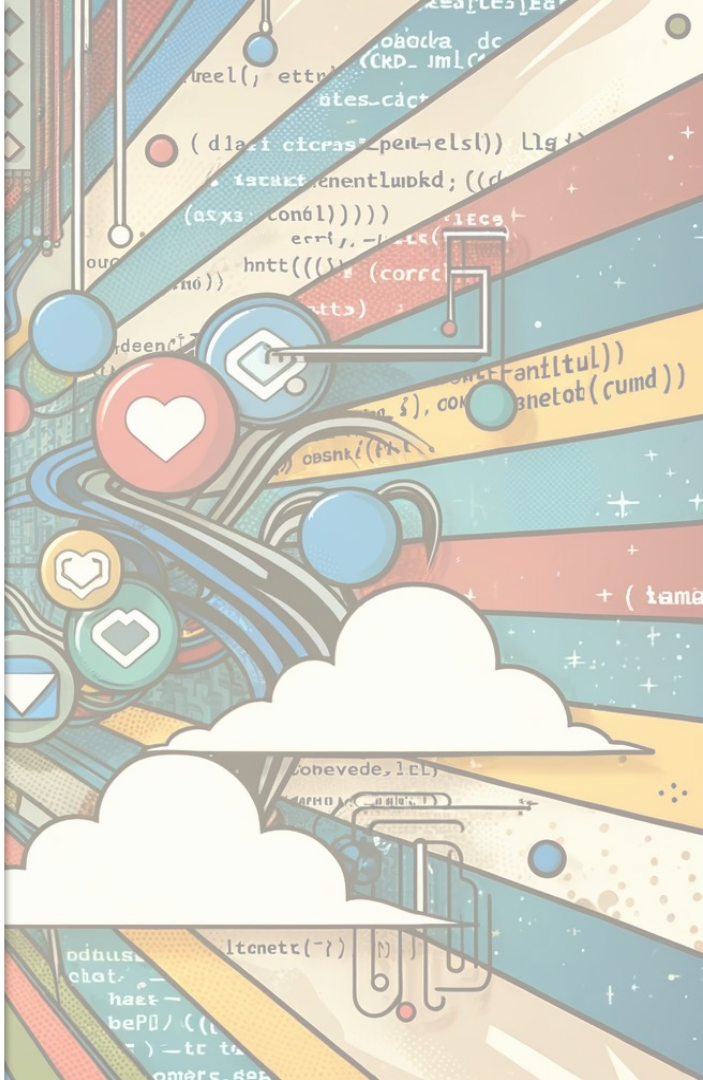
Walidacja przez middleware

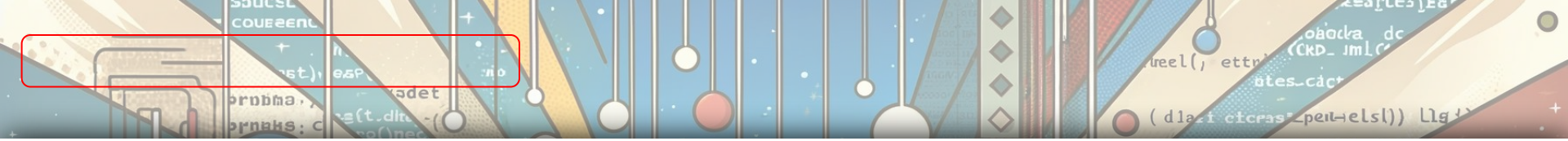
- Automatyzacja
- Czystość Kodu
- Centralizacja Walidacji

- Zależność od Frameworka
- Zarządzanie Wyjątkami
- Potencjalne opóźnienia w przetwarzaniu

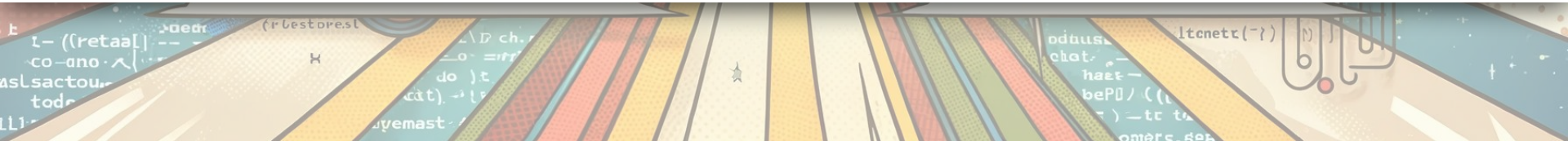
WALIDACJA REGUŁ BIZNESOWYCH

- Uczeń może zarezerwować termin lekcji u korepetytora.
- Korepetytor może mieć umówione maksymalnie 3 spotkania w tygodniu*
- *chyba, że jego ocena w serwisie przekracza 3 gwiazdki

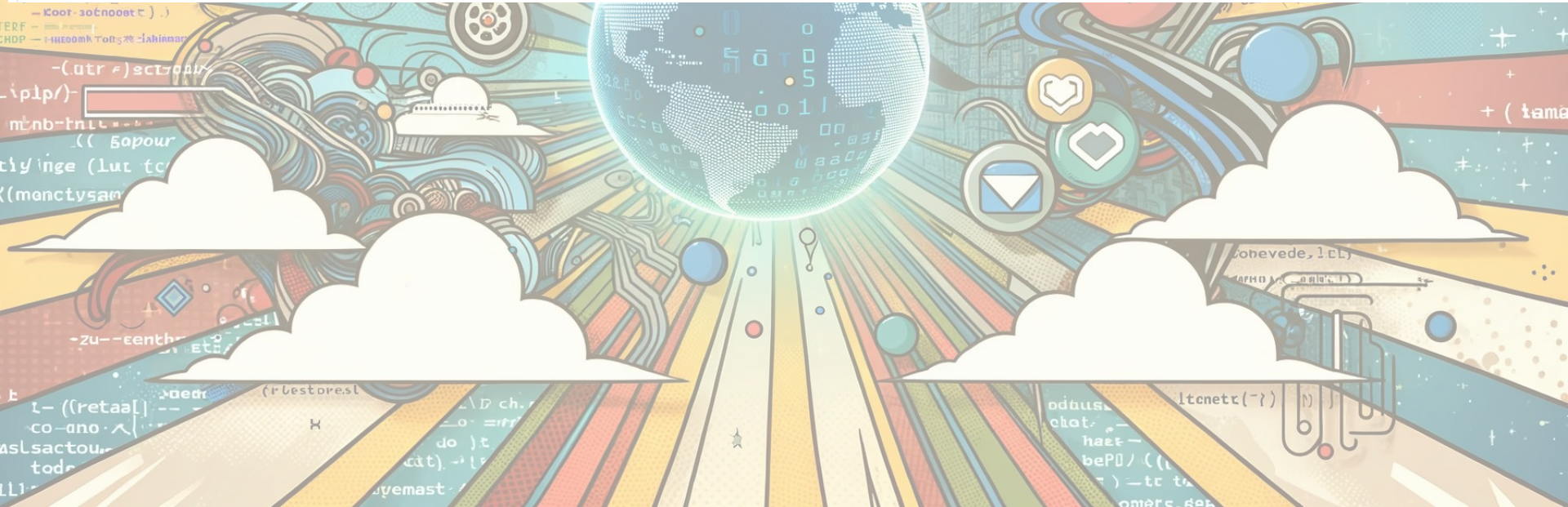




```php
public function __construct(
    #[Assert\Length(min: 3)]
    public string $subject,
    #[Assert\GreaterThan(value: 'today')]
    public \DateTimeImmutable $startDate,
    #[Assert\LessThanOrEqual(value: 120)]
    public int $duration,
    public string $teacherId,
    public string $studentId,
) {
    $this->endDate = $this->startDate->modify(sprintf( format: '+%d minutes', $duration));
}
```



```php
class TeacherLessonLimit extends Constraint
{
    public string $message = 'Teacher {{ teacherId }} has reached the limit of lessons per day';
    public function getTargets(): string
    {
        return self::CLASS_CONSTRAINT;
    }
}
```

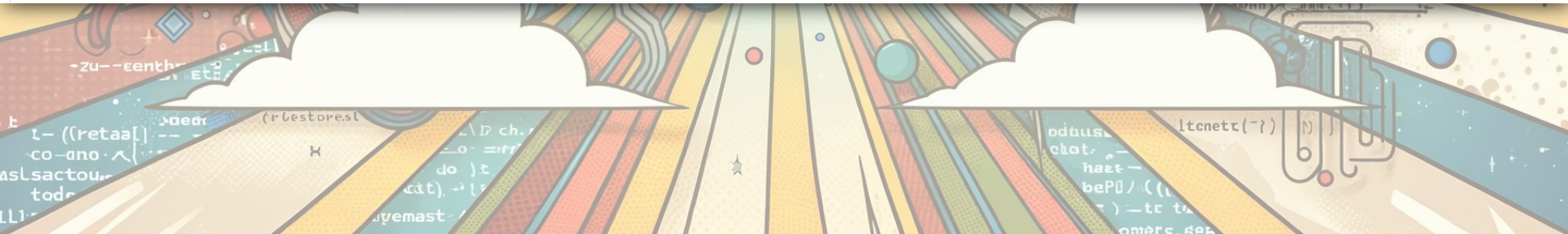


```php
public function validate(mixed $value, Constraint $constraint): void
{
    if (!$constraint instanceof TeacherLessonLimit) {
        throw new UnexpectedTypeException($constraint, expectedType: TeacherLessonLimit::class);
    }
    if (!$value instanceof CreateLessonCommand) {
        throw new UnexpectedTypeException($value, expectedType: CreateLessonCommand::class);
    }

    $lessons = $this->lessonRepository->findByTeacherIdAndDate($value->teacherId, $value->startDate);
    $rating = $this->ratingRepository->findByTeacherId($value->teacherId);

    if ($rating < 3 && count($lessons) > 3) {
        $this->context->buildViolation($constraint->message)
            ->setParameter('{{ teacherId }}', $value->teacherId)
            ->addViolation();
    }
}
```

```php
public function execute(
    CreateLesson $createLesson
): void {

    $teacher = $this->teacherRepository->find($createLesson->getTeacherId());
    $teacherLessons = $this->lessonRepository->findByTeacherIdAndDate(
        $createLesson->getTeacherId(),
        $createLesson->getStartDate()
    );

    if ($teacher->getRating() < 3 && count($teacherLessons) > 3) {
        throw new \LogicException( message: 'Teacher cannot schedule lesson');
    }

    //...
```

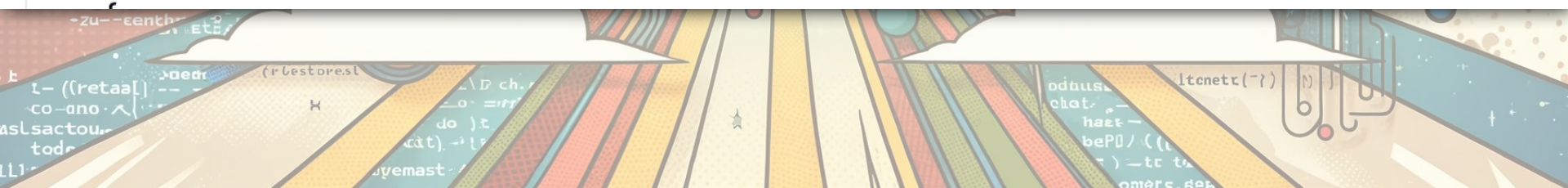

DRY

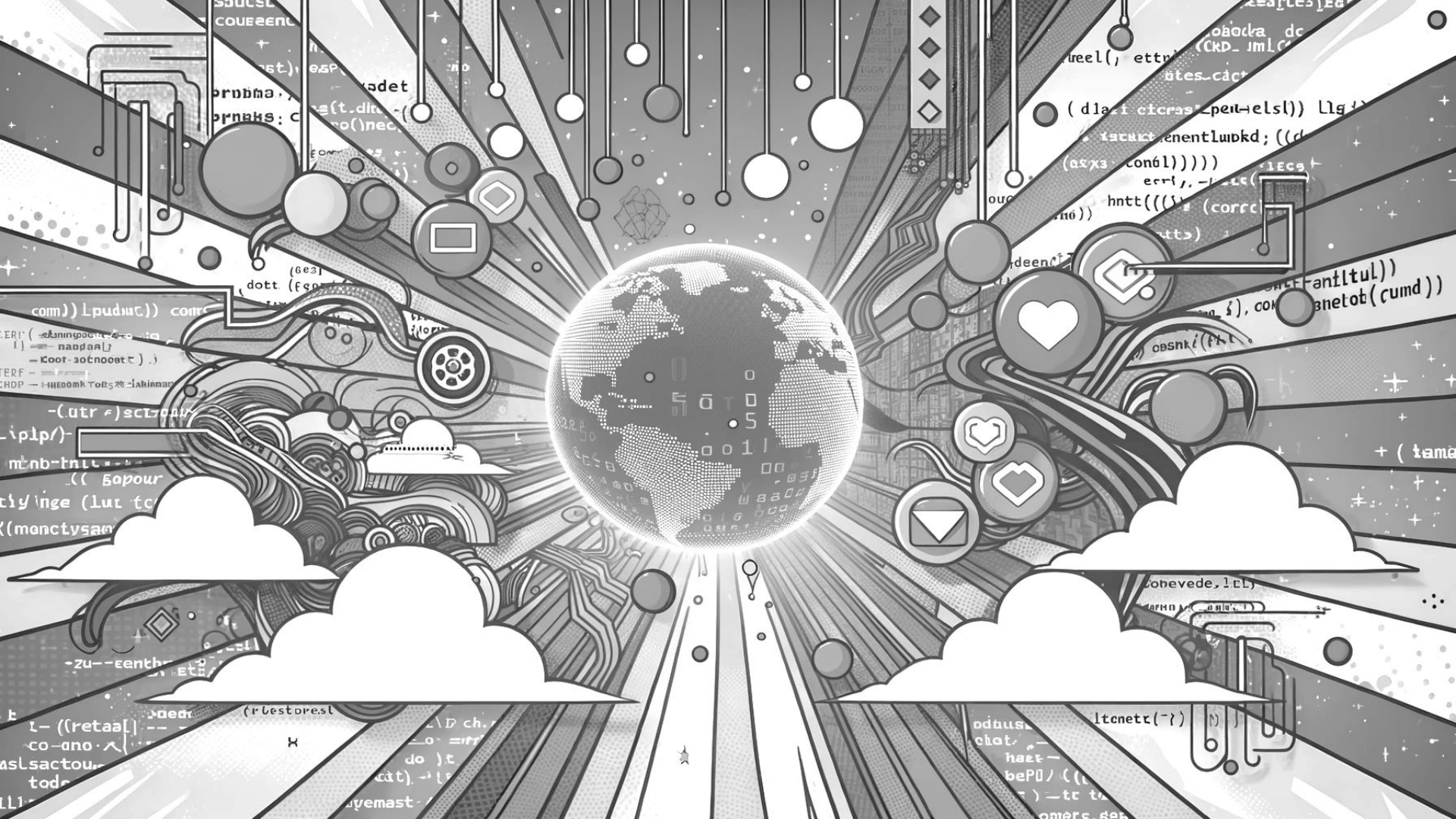
```php
class CanScheduleTeacherLessonService
{
    public function execute(string $teacherId, \DateTimeImmutable $date): bool
    {
        $teacher = $this->teacherRepository->find($teacherId);
        $lessons = $this->lessonRepository->findByTeacherIdAndDate($teacherId, $date);
        return $teacher->getRating() >= 3 || count($lessons) <= 3;
    }
}
```



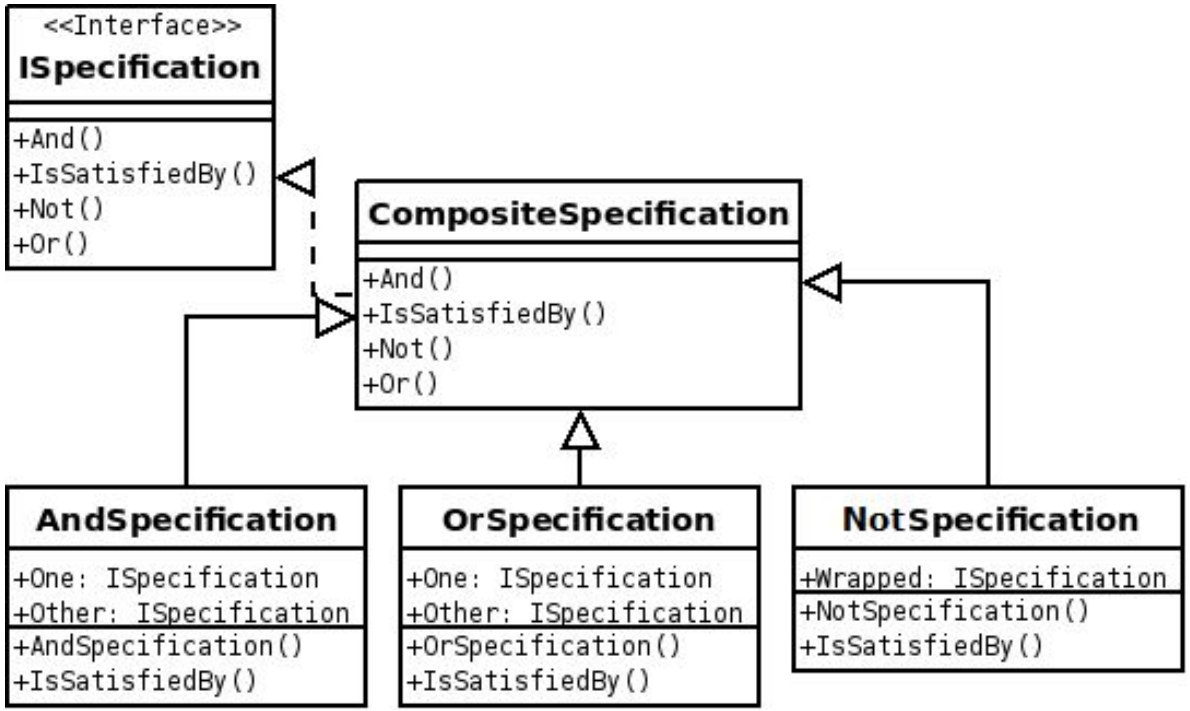
```php
class TeacherLessonLimitValidator extends ConstraintValidator
{
    public function __construct(
        private CanScheduleTeacherLessonService $canScheduleTeacherLessonService,
    ) {    You, Moments ago • Uncommitted changes
    }

    public function validate(mixed $value, Constraint $constraint): void
    {
        if (!$constraint instanceof TeacherLessonLimit) {...}
        if (!$value instanceof CreateLessonCommand) {...}

        if ($this->canScheduleTeacherLessonService->execute($value->teacherId)) {
            $this->context->buildViolation($constraint->message)
                ->setParameter('{{ teacherId }}', $value->teacherId)
                ->addViolation();
        }
    }
```

```php
public function execute(
    CreateLesson $createLesson,
    CanScheduleTeacherLessonService $canScheduleTeacherLessonService
): void {

    if (!$canScheduleTeacherLessonService->execute($teacher)) {
        throw new \LogicException( message: 'Teacher cannot schedule lesson');
    }
```





Specification



```php
class CanScheduleTeacherLesson extends Specification
{
    public function __construct(
        private readonly LessonRepository $lessonRepository,
        private readonly \DateTimeImmutable $date
    ) {
    }

    public function isSatisfiedBy(Teacher $teacher): bool
    {
        $lessons = $this->lessonRepository->findByTeacherIdAndDate($teacher->getId(), $this->date);
        return $teacher->getRating() >= 3 || count($lessons) <= 3;
    }
}
```

```php
class AndSpecification extends Specification
{
    public function __construct(
        private readonly Specification $left,
        private readonly Specification $right
    ) {
    }

    public function isSatisfiedBy(Teacher $teacher): bool
    {
        return $this->left->isSatisfiedBy($teacher)
            && $this->right->isSatisfiedBy($teacher);
    }
}
```

```php
public function execute(
    CreateLesson $createLesson,
    CanScheduleTeacherLessonSpecification $canScheduleTeacherLessonSpecification
): void {
    $teacher = $this->teacherRepository->find($createLesson->teacherId);

    if (!$canScheduleTeacherLessonSpecification->isSatisfiedBy($teacher)) {
        throw new \LogicException( message: 'Teacher cannot schedule lesson');
    }

    // ...
}
```

Specyfikacje

- Jasna Ekspresja Reguł
- Kompozycja Deklaratywna
- Zachęta do Myślenia na Poziomie Reguł Biznesowych
- Przenośność

Dane wejściowe — podsumowanie

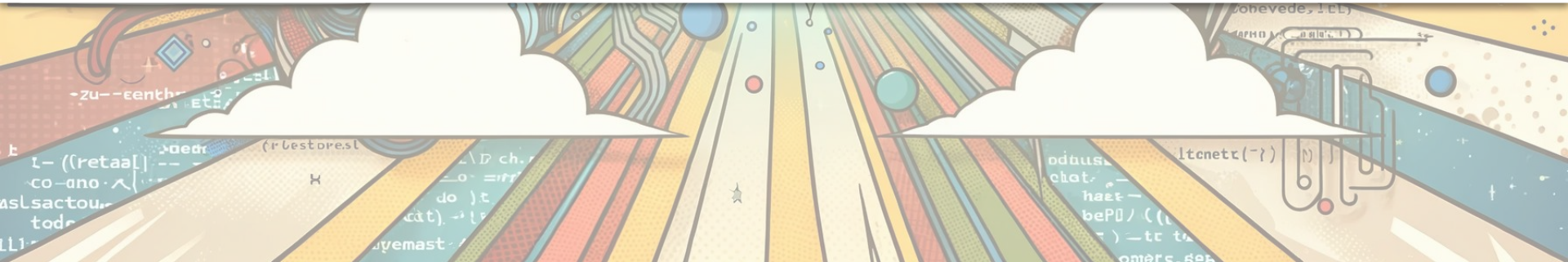
- Odchodzenie od bezpośredniego Używania Requestu
- Tworzenie Obiektów DTO
  - Implementacja Własnej Fabryki
  - Mechanizm Mapowania Symfony
- Wykorzystanie DTO jako Command i Wzorzec CQS
- Wsparcie Symfony w Walidacji
- Różne Metody Uruchamiania Walidacji
  - Automatyczne przy Mapowaniu:
  - Ręczne Uruchamianie w Fabrykach
  - Middleware w Messengerze
- Centralizacja Powtarzającej się Logiki
- Wykorzystanie Wzorca Specification

Dane wejściowe — podsumowanie

- Odchodzenie od bezpośredniego Używania Requestu
- Tworzenie Obiektów DTO
  - Implementacja Własnej Fabryki
  - Mechanizm Mapowania Symfony
- Wykorzystanie DTO jako Command i Wzorzec CQS
- Wsparcie Symfony w Walidacji
- Różne Metody Uruchamiania Walidacji
  - Automatyczne przy Mapowaniu:
  - Walidator wstrzykiwany do fabryk
  - Middleware w Messengerze
- Centralizacja Powtarzającej się Logiki
- Wykorzystanie Wzorca Specification

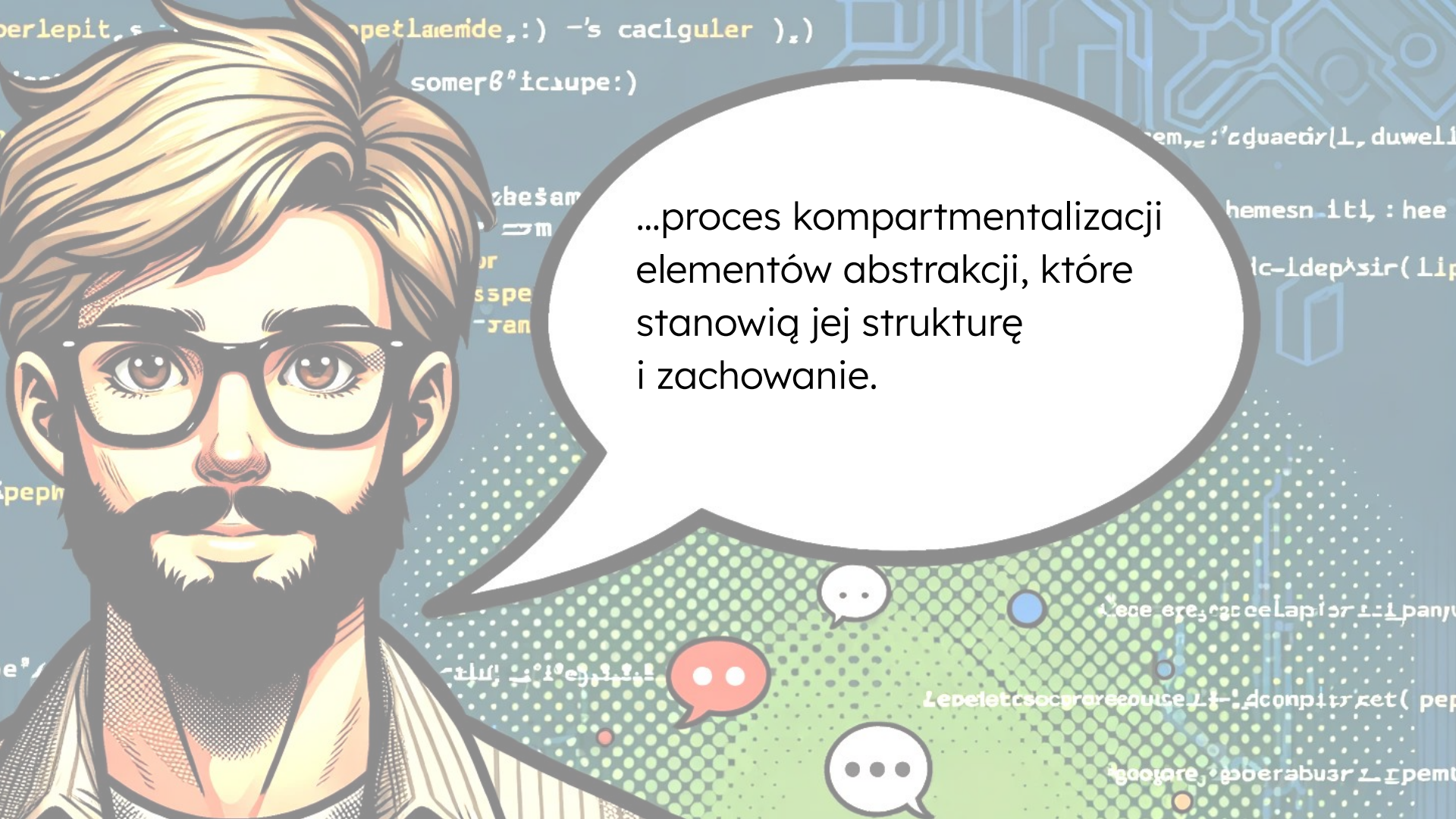

ENTITY

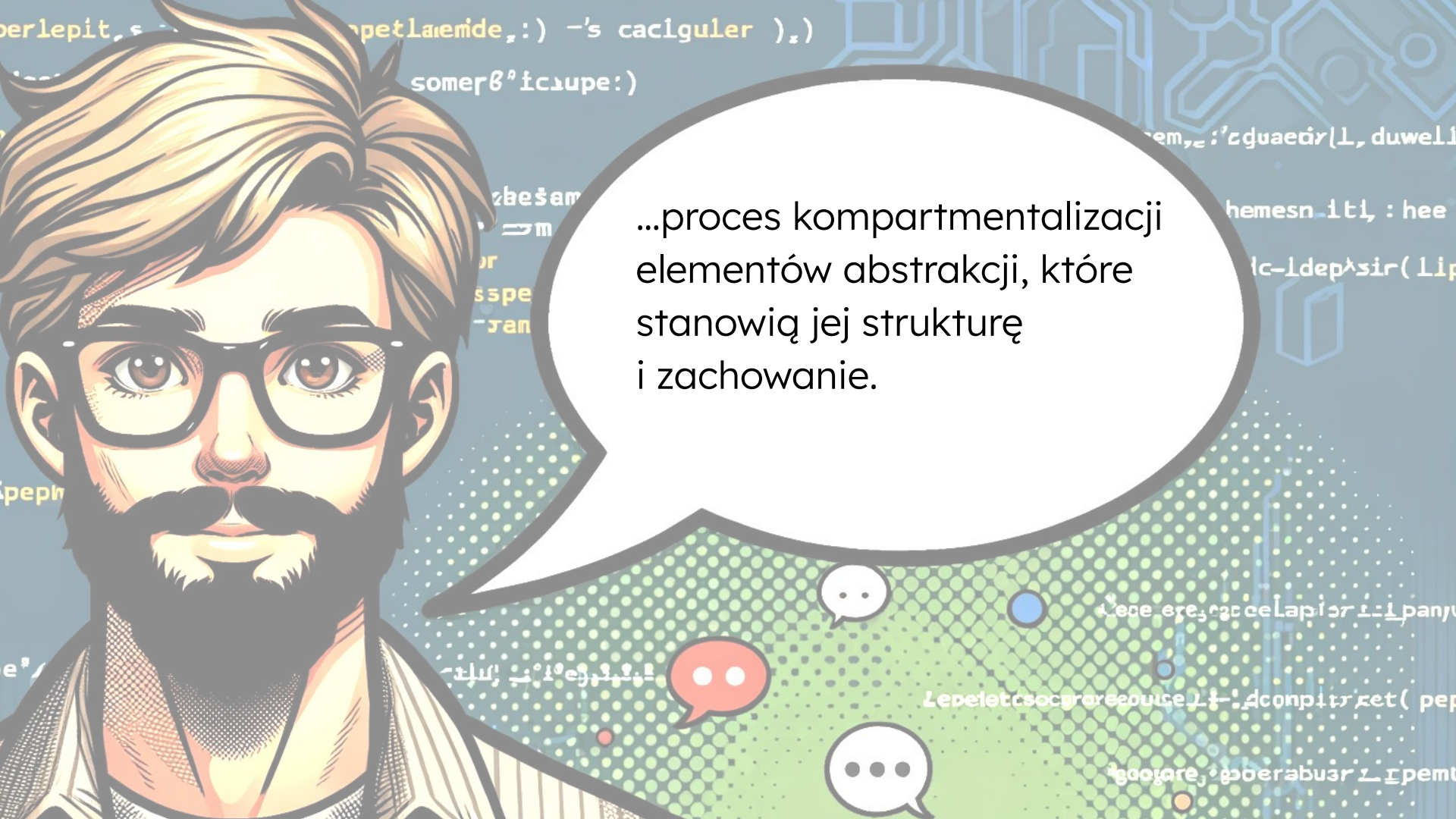
Czy programujemy obiektowo?

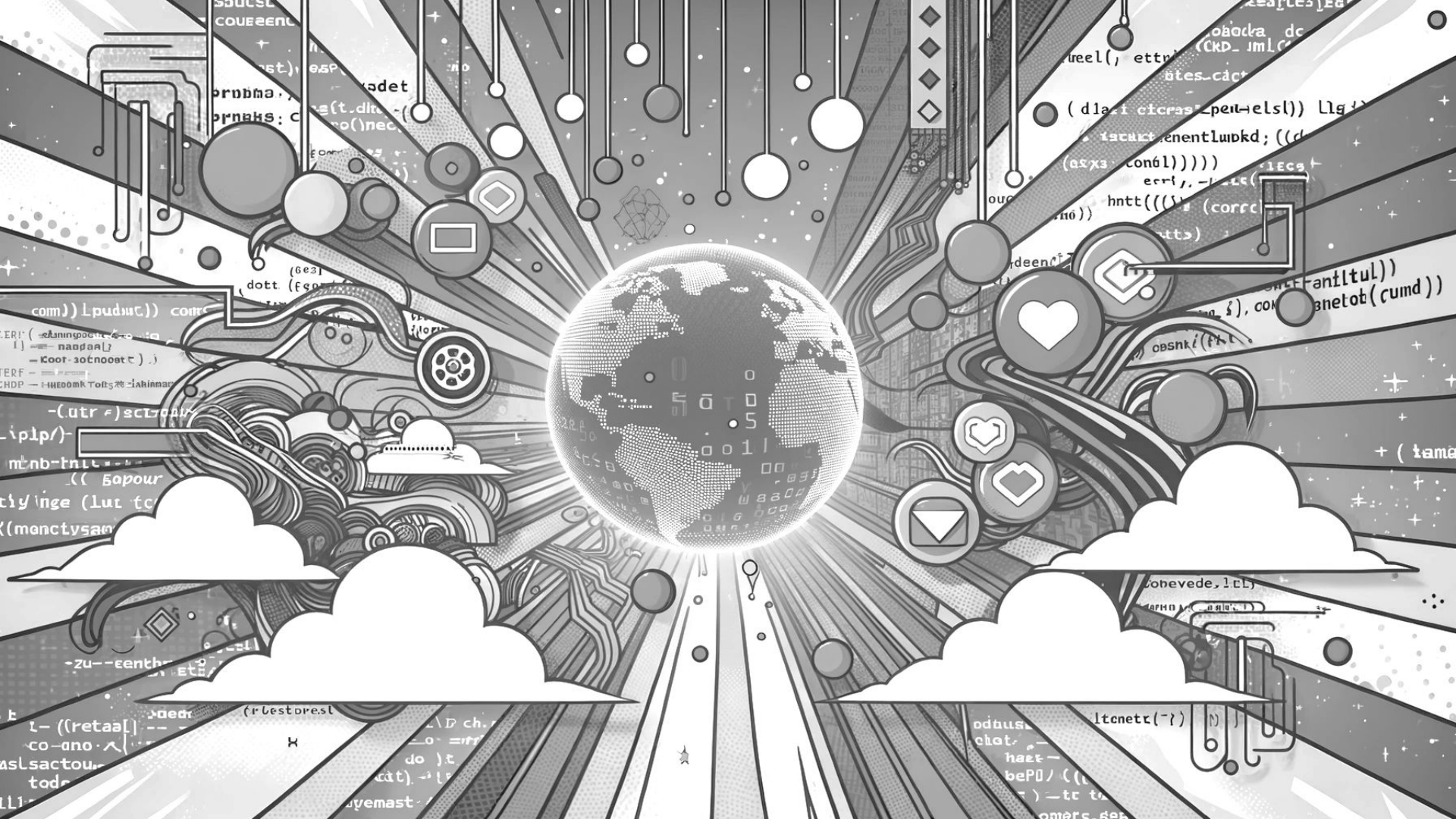


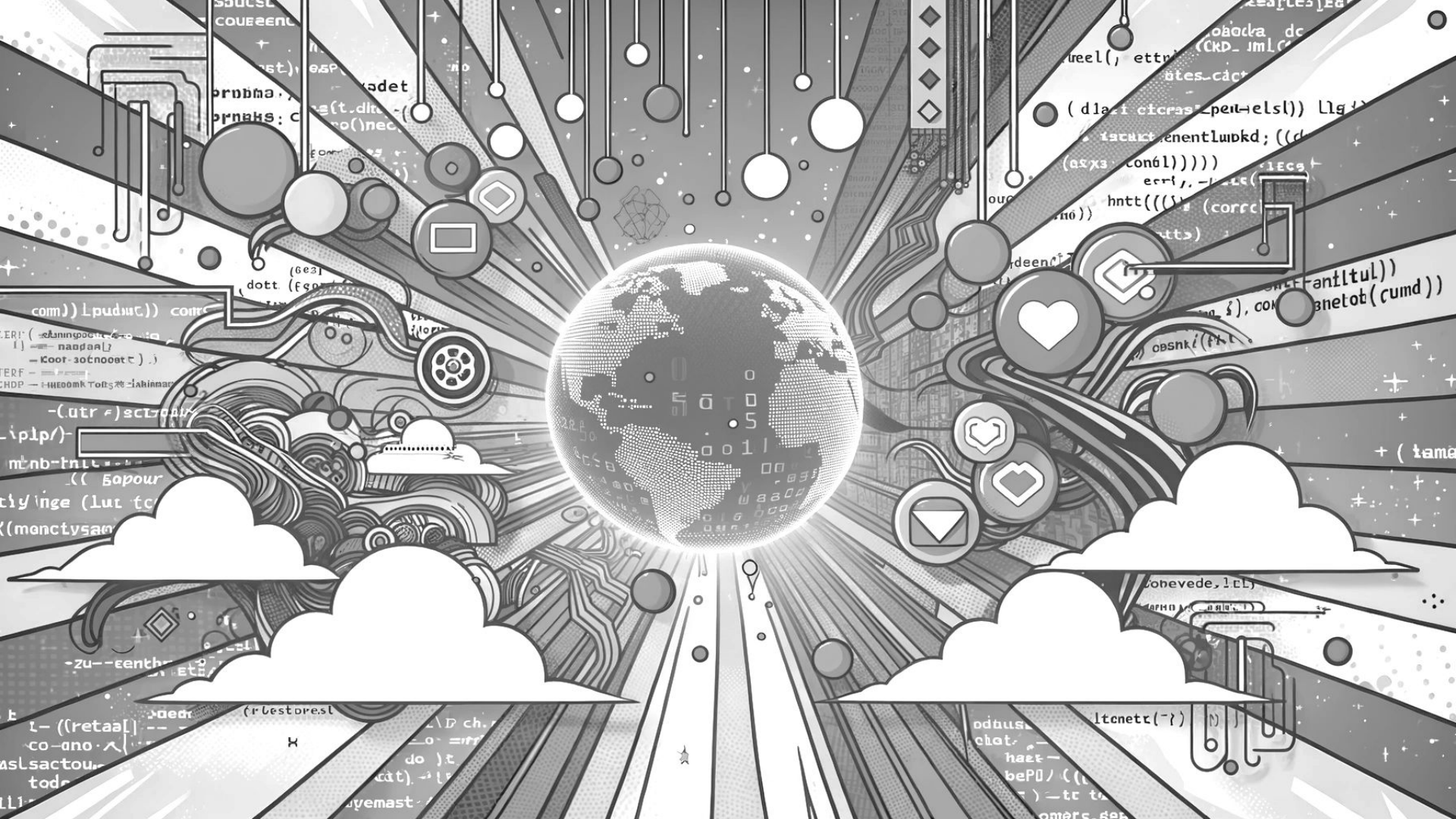
```php
#[ORM\Entity]
#[ORM\Table(name: "lessons")]
class LessonEntity
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column(type: "integer")]
    private int $id;

    #[Assert\Length(min: 3)]
    private string $subject;

    #[ORM\Column(type: "datetime")]
    private \DateTimeImmutable $from;

    #[ORM\Column(type: "datetime")]
    private \DateTimeImmutable $to;

    #[ORM\ManyToOne(targetEntity: Teacher::class)]
    #[ORM\JoinColumn(name: "teacher_id", referencedColumnName: "id")]
    private Teacher $teacher;

    #[ORM\ManyToOne(targetEntity: Student::class)]
    #[ORM\JoinColumn(name: "student_id", referencedColumnName: "id")]
    private Student $student;
```

```php
public function getId(): int
{
    return $this->id;
}

public function setId(int $id): LessonEntity
{
    $this->id = $id;
    return $this;
}

public function getSubject(): Subject
{
    return $this->subject;
}

public function setSubject(Subject $subject): LessonEntity
{
    $this->subject = $subject;
    return $this;
}

public function getPeriod(): Period
{
    return $this->period;
}

public function setPeriod(Period $period): LessonEntity
{
    $this->period = $period;
    return $this;
}

public function getTeacher(): Teacher
{
    return $this->teacher;
}

public function setTeacher(Teacher $teacher): LessonEntity
```


...proces kompartmentalizacji elementów abstrakcji, które stanowią jej strukturę i zachowanie.


...proces kompartmentalizacji elementów abstrakcji, które stanowią jej strukturę i zachowanie.





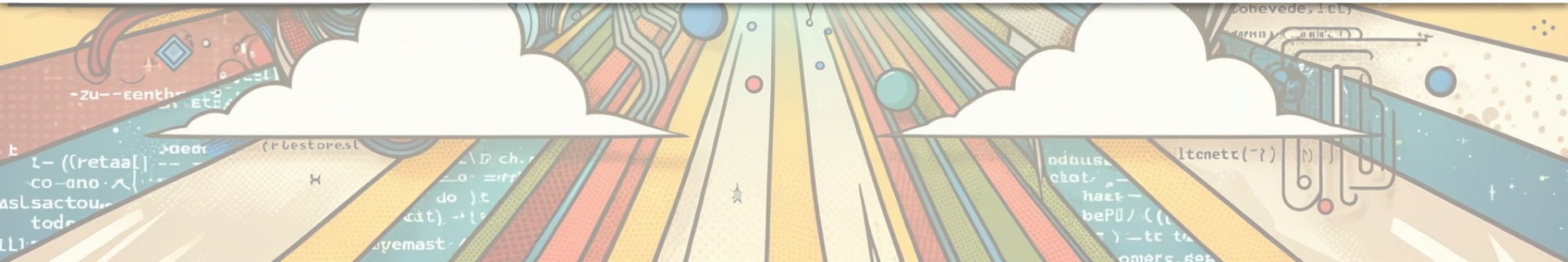
Implementacja

```php
public function execute(
    CreateLesson $createLesson,
    CanScheduleTeacherLessonSpecification $canScheduleTeacherLessonSpecification
): void {
    $teacher = $this->teacherRepository->find($createLesson->teacherId);

    $lesson = Lesson::create(
        $teacher,
        $student,
        $subject,
        $dateFrom,
        $dateTo,
        $canScheduleTeacherLessonSpecification
    );


    // ...
}
```

```php
public static function create(
    Teacher $teacher,
    Student $student,
    Subject $subject,
    DateTime $startTime,
    DateTime $endTime,
    CanScheduleTeacherLessonSpecification $canScheduleTeacherLessonSpecification,
): self {
    if (!$canScheduleTeacherLessonSpecification->isSatisfiedBy($teacher)) {
        throw new \LogicException(sprintf(
            format: 'Teacher %s has reached the limit of lessons per day',
            $teacher->getId()
        ));
    }

    // ...

    return new self($teacher, $student, $subject, $startTime, $endTime);
}
```

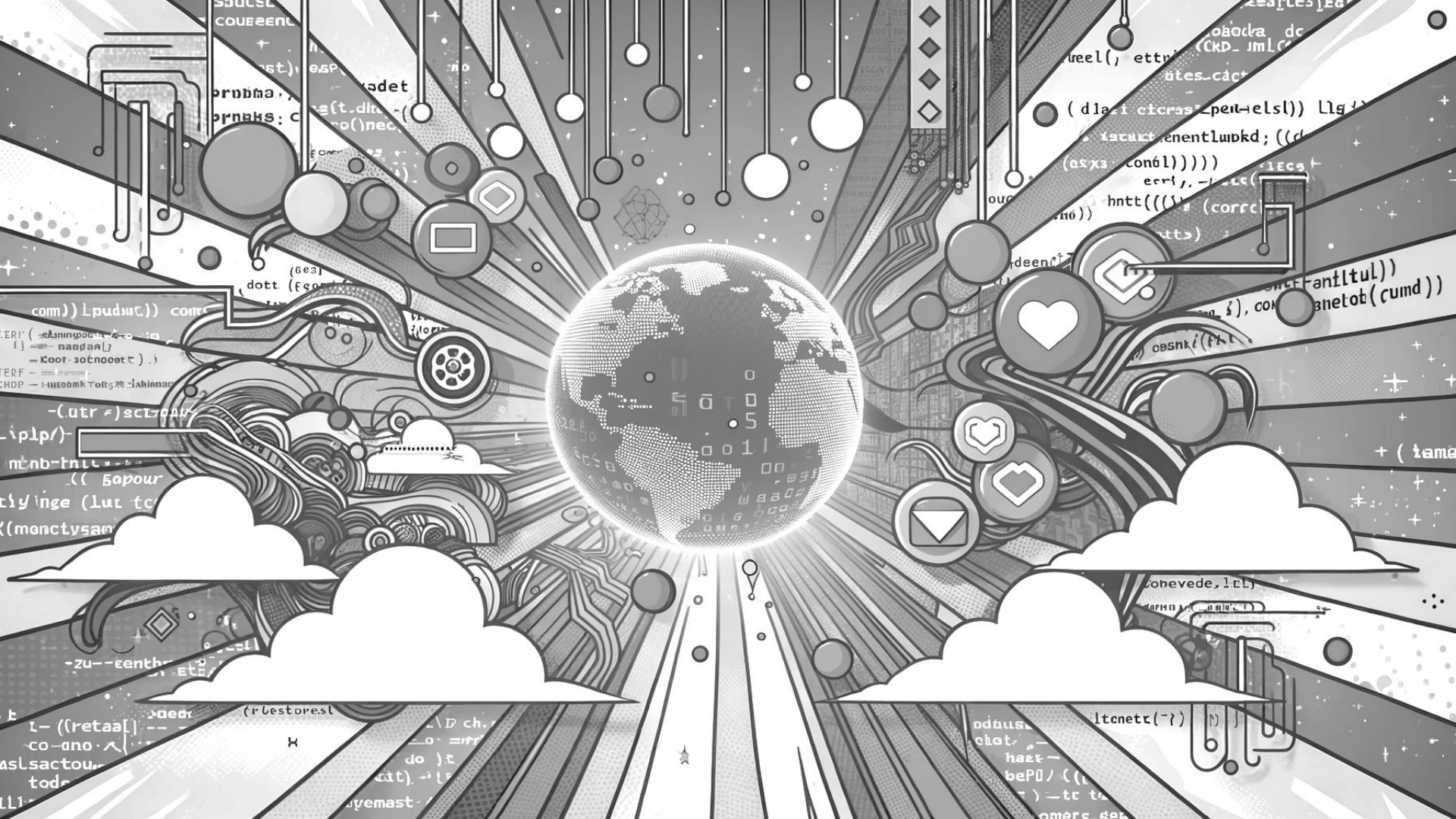
```php
public static function create(
    Teacher $teacher,
    Student $student,
    Subject $subject,
    DateTime $startTime,
    DateTime $endTime,
    CanScheduleTeacherLessonSpecification $canScheduleTeacherLessonSpecification,
): self {
    if (!$canScheduleTeacherLessonSpecification->isSatisfiedBy($teacher)) {
        throw new \LogicException(sprintf(
            format: 'Teacher %s has reached the limit of lessons per day',
            $teacher->getId()
        ));
    }

    // ...

    return new self($teacher, $student, $subject, $startTime, $endTime);
}
```

Mapowanie

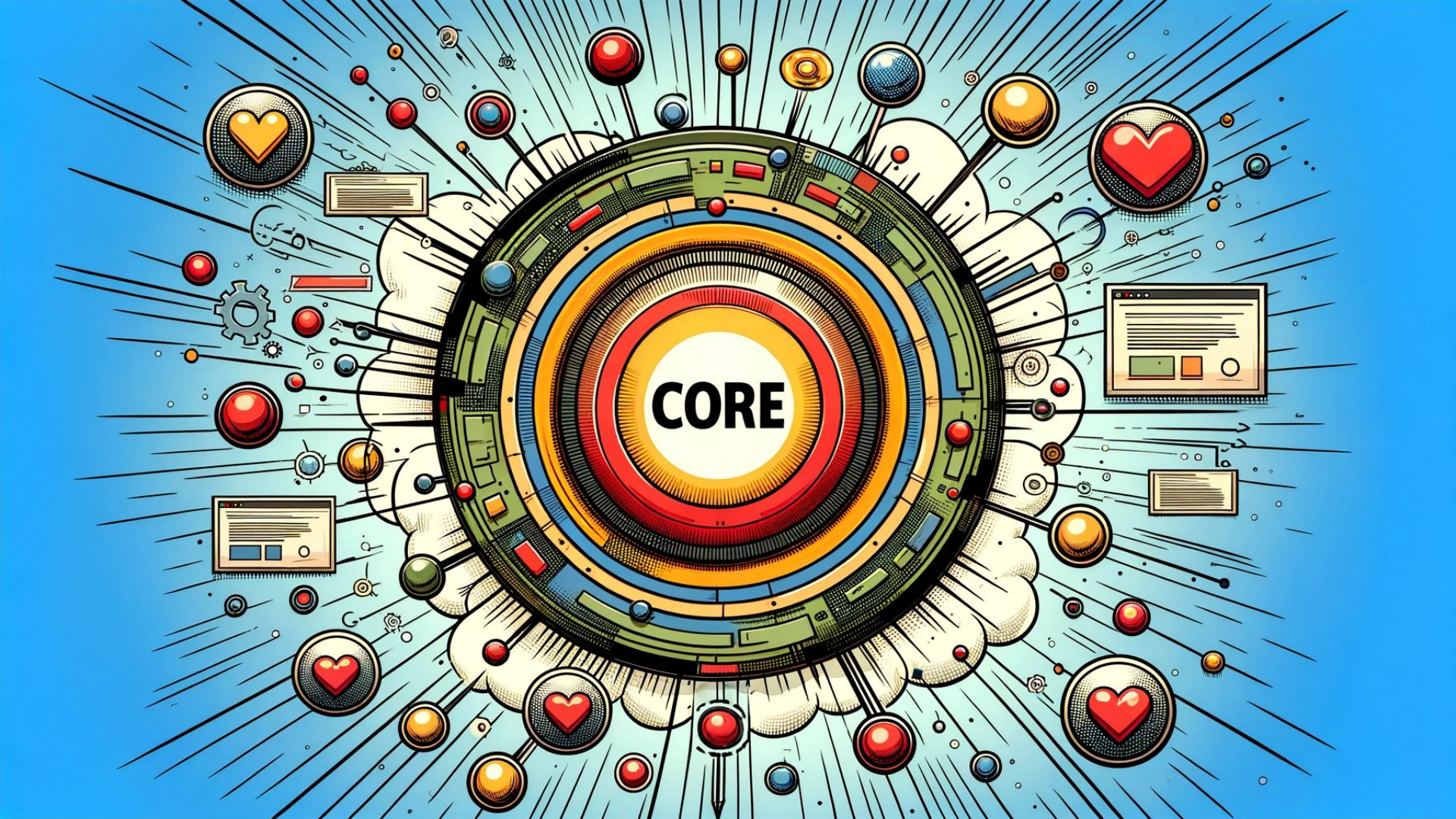


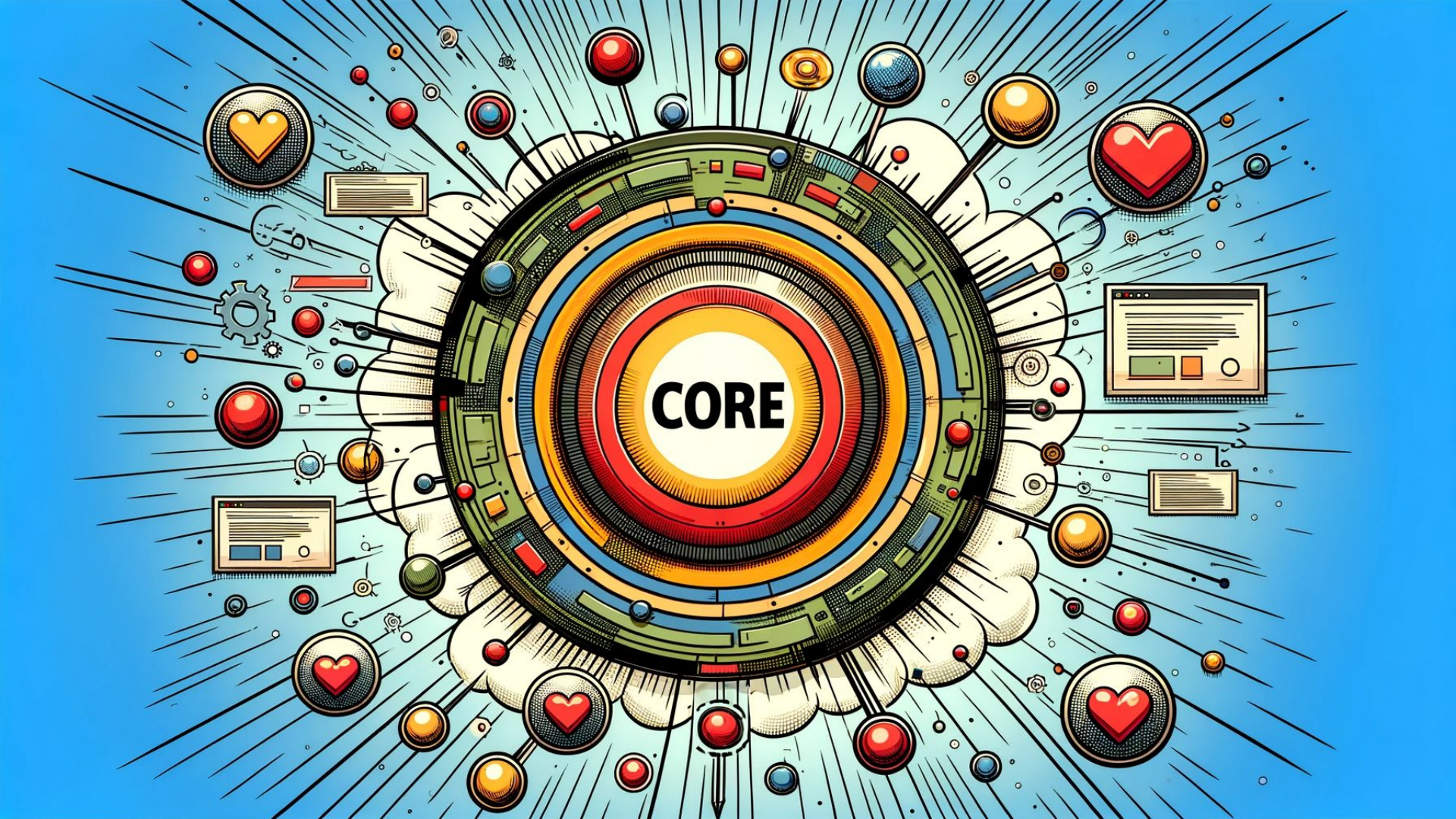
```php
#[ORM\Table(name: "lessons")]
class LessonEntity
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column(type: "integer")]
    private int $id;

    #[ORM\Column(type: "string")]
    private string $subject;

    #[ORM\Column(type: "datetime")]
    private \DateTimeImmutable $from;

    #[ORM\Column(type: "datetime")]
    private \DateTimeImmutable $to;

    #[ORM\ManyToOne(targetEntity: Teacher::class)]
    #[ORM\JoinColumn(name: "teacher_id", referencedColumnName
    private Teacher $teacher;

    #[ORM\ManyToOne(targetEntity: Student::class)]
```

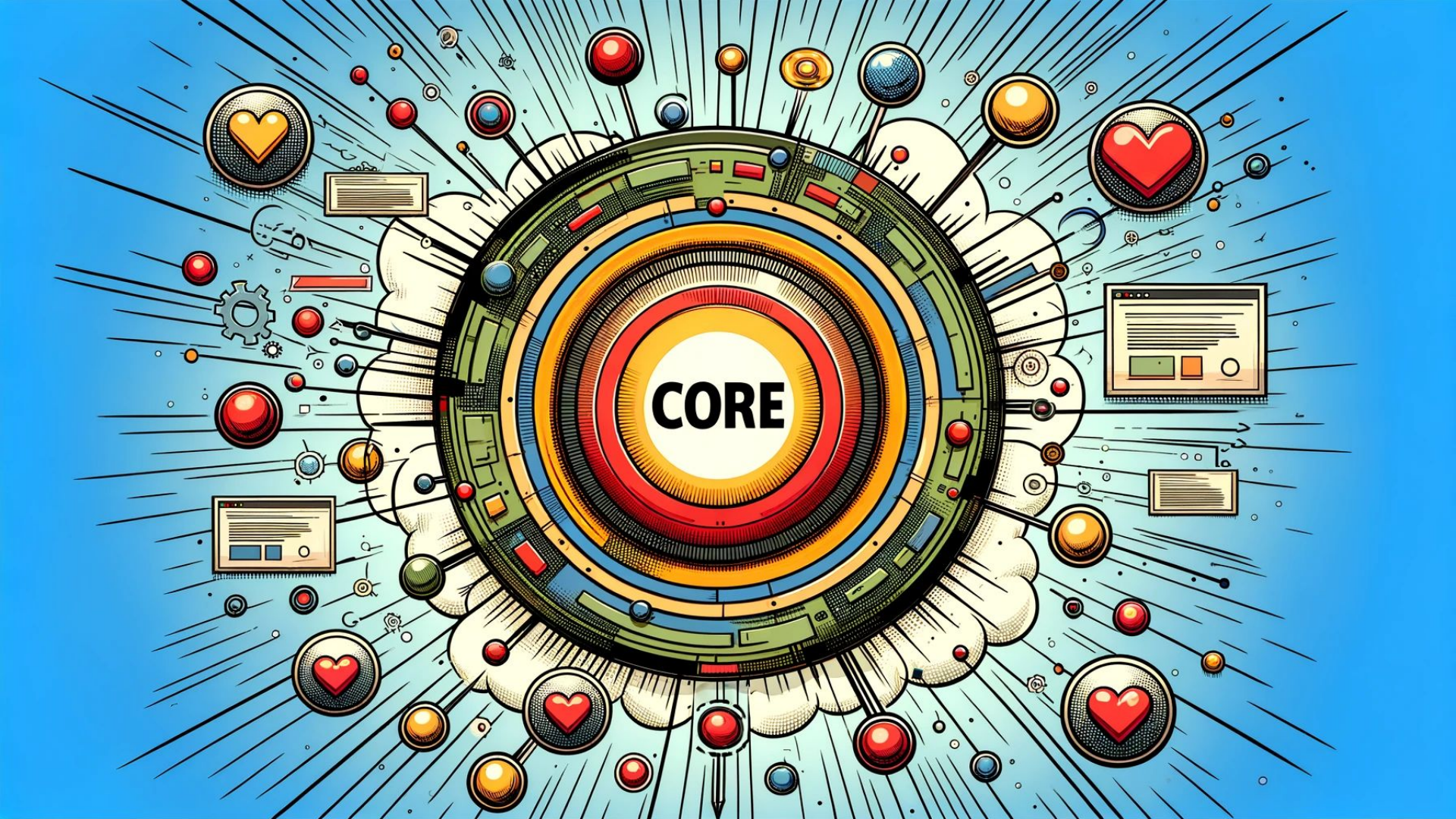
You, 12 minutes ag



```php
class LessonRepository extends ServiceEntityRepository
{
    public function find(string $id): Lesson
    {
        $dataEntity = $this->find($id);
        return Lesson::fromDto($dataEntity->getAsDto());
    }

    public function save(Lessno $id): void
    {
        $dataEntity = LessonOrmEntity::fromDto($lesson->getAsDto());
        $this->entityManager->persist($dataEntity);
        $this->entityManager->flush();
    }
}
```


CORE


CORE

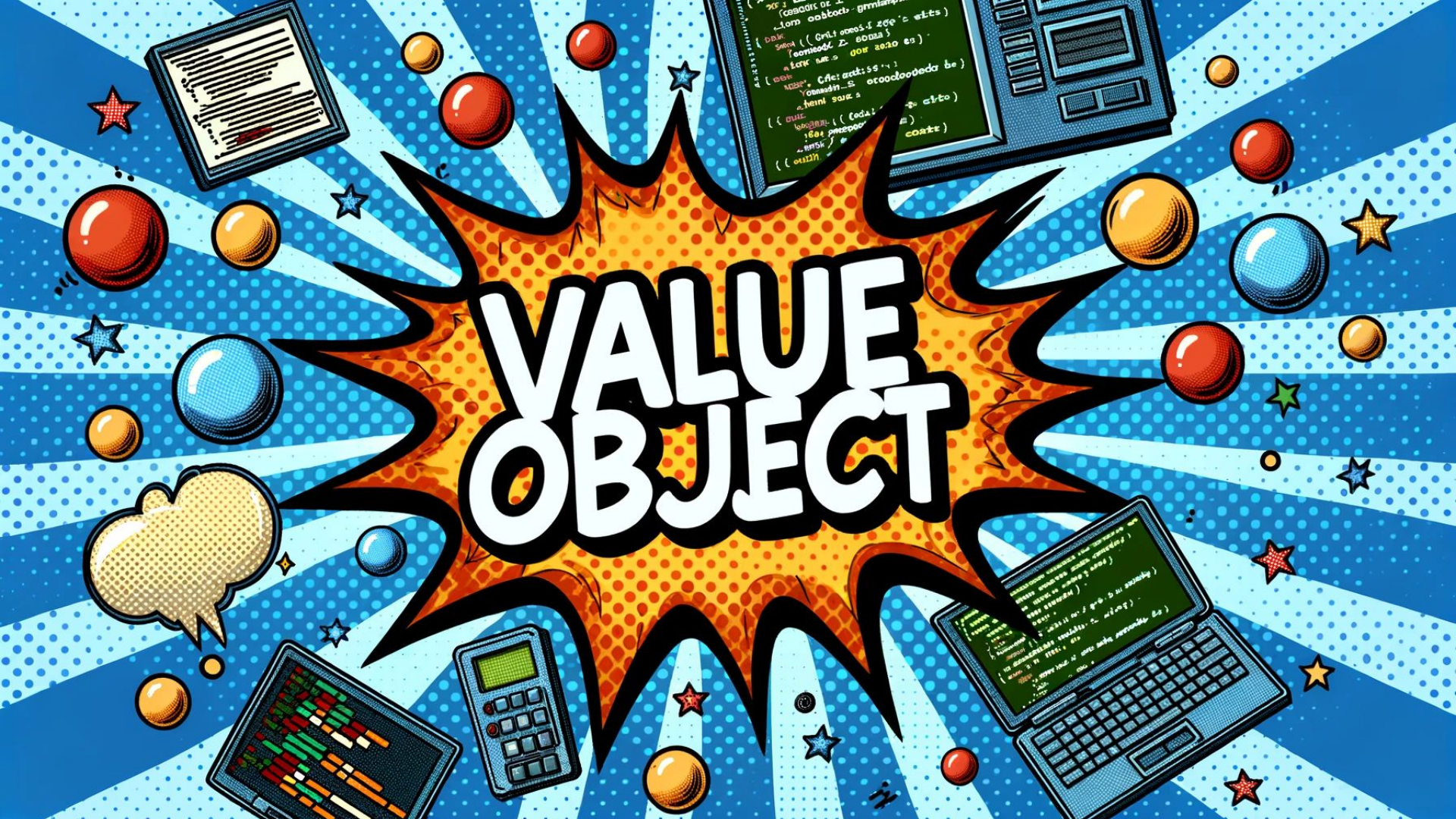

CORE

Encje — podsumowanie

- Encja – Więcej niż Struktura Danych
- Znaczenie Hermetyzacji
- Modelowanie Zachowań Przed Danymi
- Przenoszenie Logiki do Encji
- Mapowanie ORM
- Rozdzielenie Encji Domenowych od Persystencyjnych

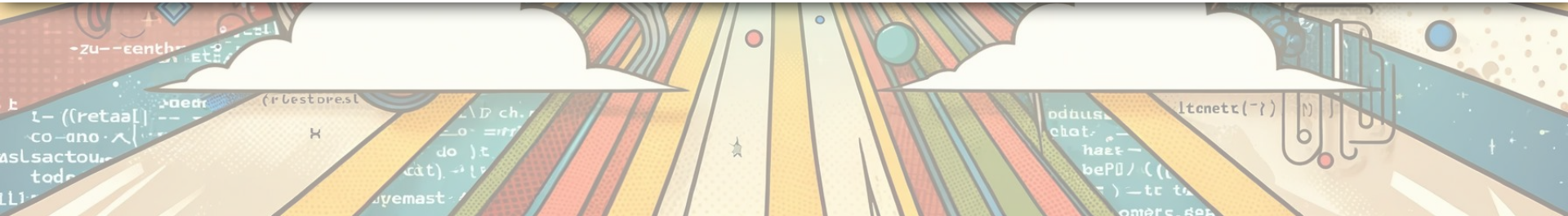
Encje — podsumowanie

- Encja – Więcej niż Struktura Danych
- Znaczenie Hermetyzacji
- Modelowanie Zachowań Przed Danymi
- Przenoszenie Logiki do Encji
- Mapowanie ORM
- Rozdzielenie Encji Domenowych od Persystencyjnych


VALUE
OBJECT

```php
#[Assert\Length(min: 3)]
private Subject $subject;
```

```php
if ($dataFrom > $currentDate->getDateTime()) {
    throw new \InvalidArgumentException( message: 'Lesson cannot be scheduled in the past');
}
if ($dataFrom > $dateTo) {
    throw new \InvalidArgumentException( message: 'Start date cannot be greater than end date');
}
if ($dateFrom->diff($dateTo) > 120) {
    throw new \InvalidArgumentException( message: 'Lesson duration can not be longer than 120 minutes');
}
```

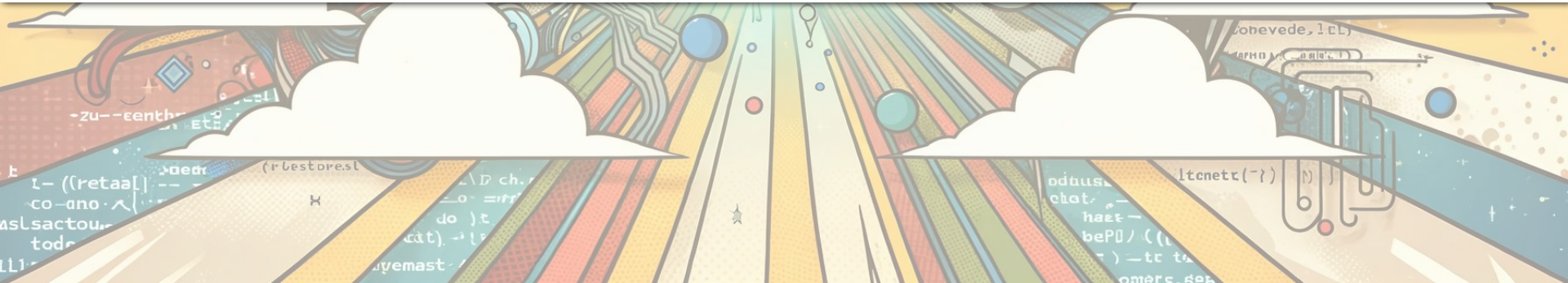


```php
class Period
{
    public function __construct(
        public readonly \DateTimeImmutable $dateFrom,
        public readonly \DateTimeImmutable $dateTo,
        private readonly TimeStamp $timeStamp
    ) {
        $this->validate();
    }

    private function validate(): void
    {
        if ($dataFrom > $currentDate->getDateTime()) {    You, 10 minutes ago • Uncommitted changes
            throw new \InvalidArgumentException( message: 'Lesson cannot be scheduled in the past');
        }
        if ($dataFrom > $dateTo) {
            throw new \InvalidArgumentException( message: 'Start date cannot be greater than end date');
        }
        if ($dateFrom->diff($dateTo) > 120) {
            throw new \InvalidArgumentException( message: 'Lesson duration can not be longer than 120 minutes');
        }
    }
```

```php
public function overlaps(Period $period): bool
{
    return $this->dateFrom= $period->dateTo && $period->dateTo >= $this->dateFrom;
}

public function isLeftContiguousWith(Period $other): bool
{
    return $this->dateFrom == $other->dateTo;
}

public function isRightContiguousWith(Period $other): bool
{
    return $this->dateTo == $other->dateFrom;
}
```

```php
class LessonPreferences
{
    public function __construct(
        public readonly TeachingStyle $teachingStyle,
        public readonly bool $preferGroupLesson,
    ) {
    }
}
```

```php
public function isEqualTo(LessonPreferences $preferences): float
{
    return $this->teachingStyle === $preferences->teachingStyle
        && $this->preferGroupLesson === $preferences->preferGroupLesson;
}
```

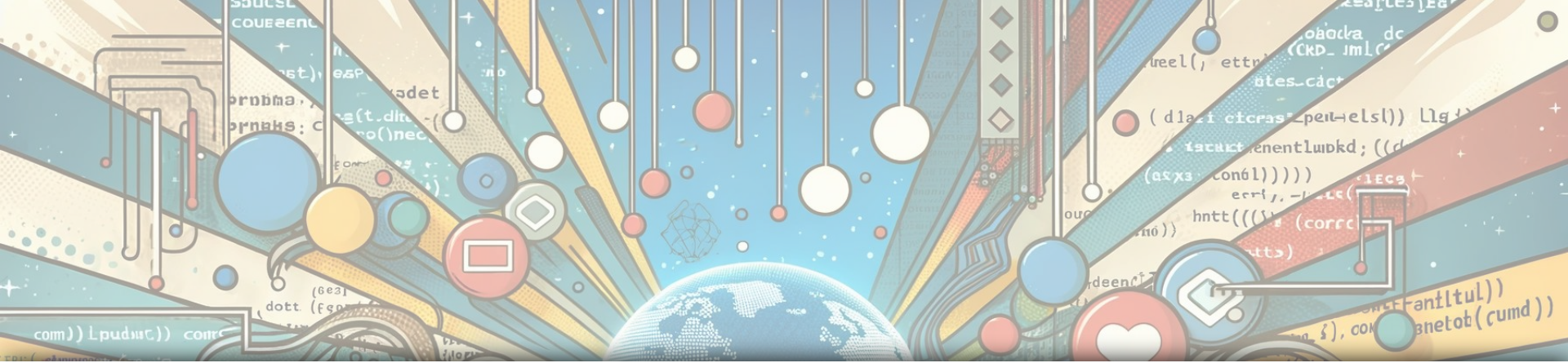


Mapowanie VO

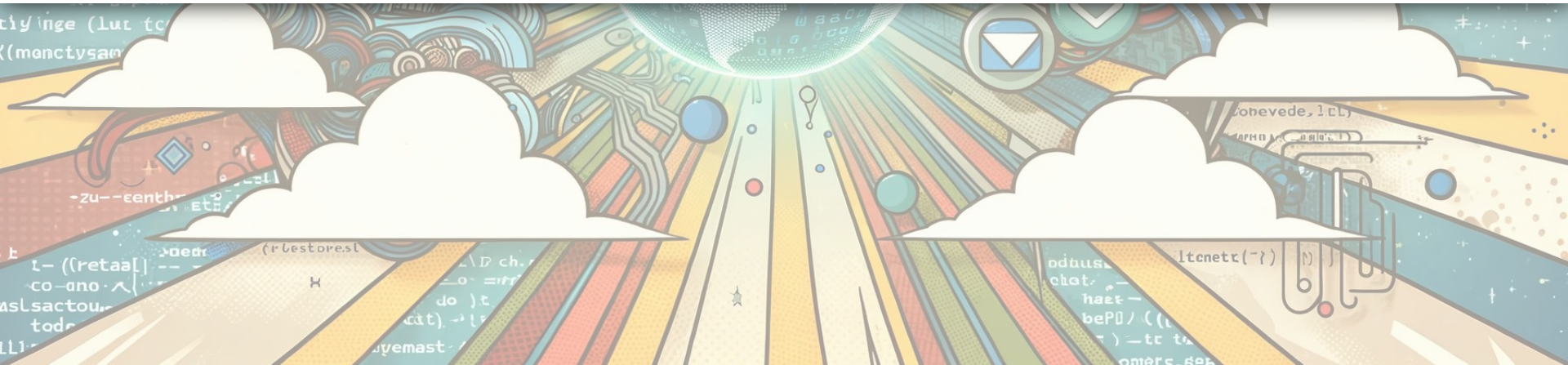
```php
class SubjectType extends StringType
{
    const SUBJECT_TYPE = 'subject_type';

    public function getName()
    {
        return self::SUBJECT_TYPE;
    }

    public function convertToPHPValue($value, AbstractPlatform $platform): Subject
    {
        if (null === $value || $value instanceof Subject) {
            return $value;
        }

        return new Subject((string)$value);
    }

    public function convertToDatabaseValue($value, AbstractPlatform $platform): string
    {
        if ($value instanceof Subject) {
            return (string)$value;
        }

        return $value;
    }
}
```

```php
#[ORM\Column(type: "subject_type")]
private Subject $subject;
```

```php
#[ORM\Embeddable]
class Period
{
    public function __construct(
        #[ORM\Column(type: "datetime")]
        public readonly \DateTimeImmutable $dateFrom,
        #[ORM\Column(type: "datetime")]
        public readonly \DateTimeImmutable $dateTo,
        private readonly TimeStamp $timeStamp
    ) {
        $this->validate();
    }
```

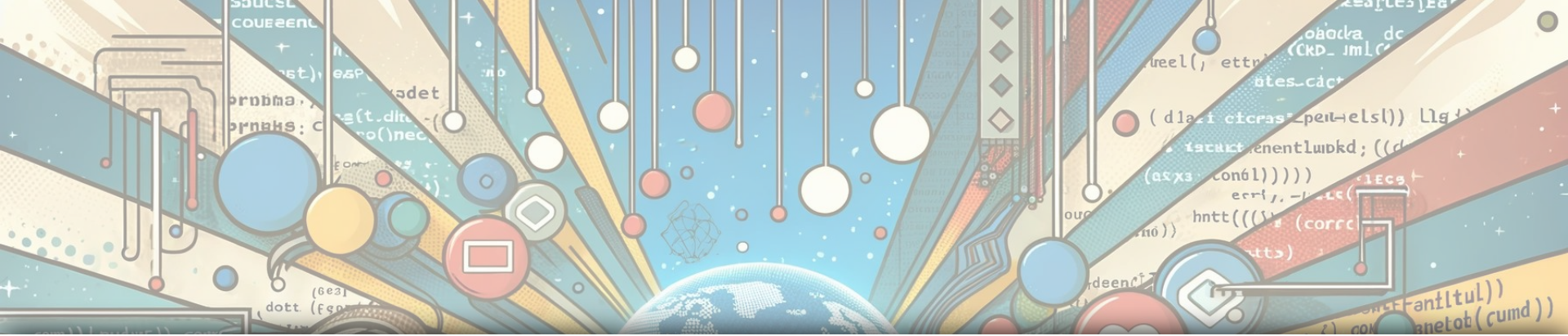


```php
#[ORM\Embedded(class: "App\ValueObject\Period", columnPrefix: false)]
private Period $period;
```

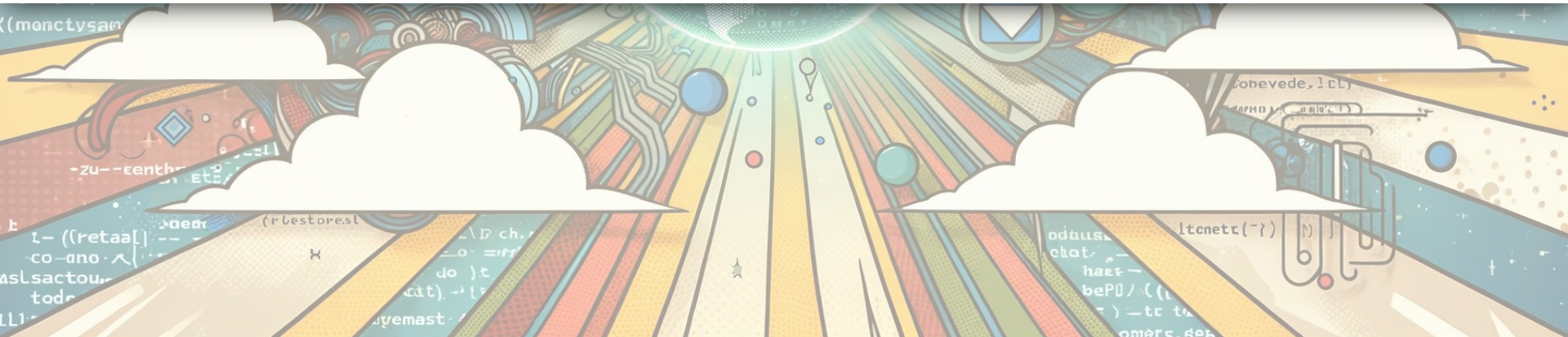


```php
class LessonPreferences
{
    public function __construct(
        public readonly TeachingStyle $teachingStyle,
        public readonly bool $preferGroupLesson,
    ) {
    }
}
```

```php
class LessonPreferences implements \JsonSerializable
{
    public function __construct(
        public readonly TeachingStyle $teachingStyle,
        public readonly bool $preferGroupLesson,
    ) {
    }

    public function jsonSerialize(): array
    {
        return [
            'teachingStyle' => $this->teachingStyle->value,
            'preferGroupLesson' => $this->preferGroupLesson,
        ];
    }
}
```



```
#[ORM\Column(type: "json")]
private LessonPreferences $lessonPreferences
```



Value Object — podsumowanie

- Lepsza Organizacja Kodu
- Ułatwienie Zarządzania
- Rozszerzanie Reguł
- Unikanie Nadmiernego Komplikowania:
- Hermetyzacja i Zarządzanie Złożoną Logiką:
- Uniwersalność:

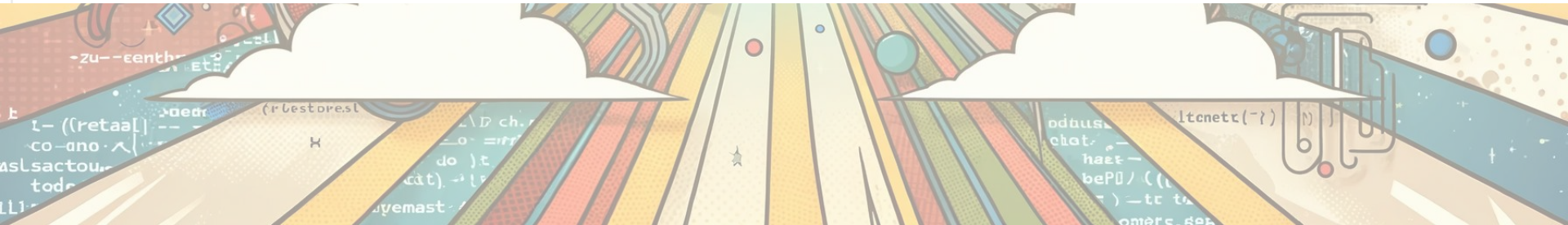
Value Object — podsumowanie

- Lepsza Organizacja Kodu
- Ułatwienie Zarządzania
- Rozszerzanie Reguł
- Unikanie Nadmiernego Komplikowania:
- Hermetyzacja i Zarządzanie Złożoną Logiką
- Uniwersalność

ODCZYT/ZAPIS...

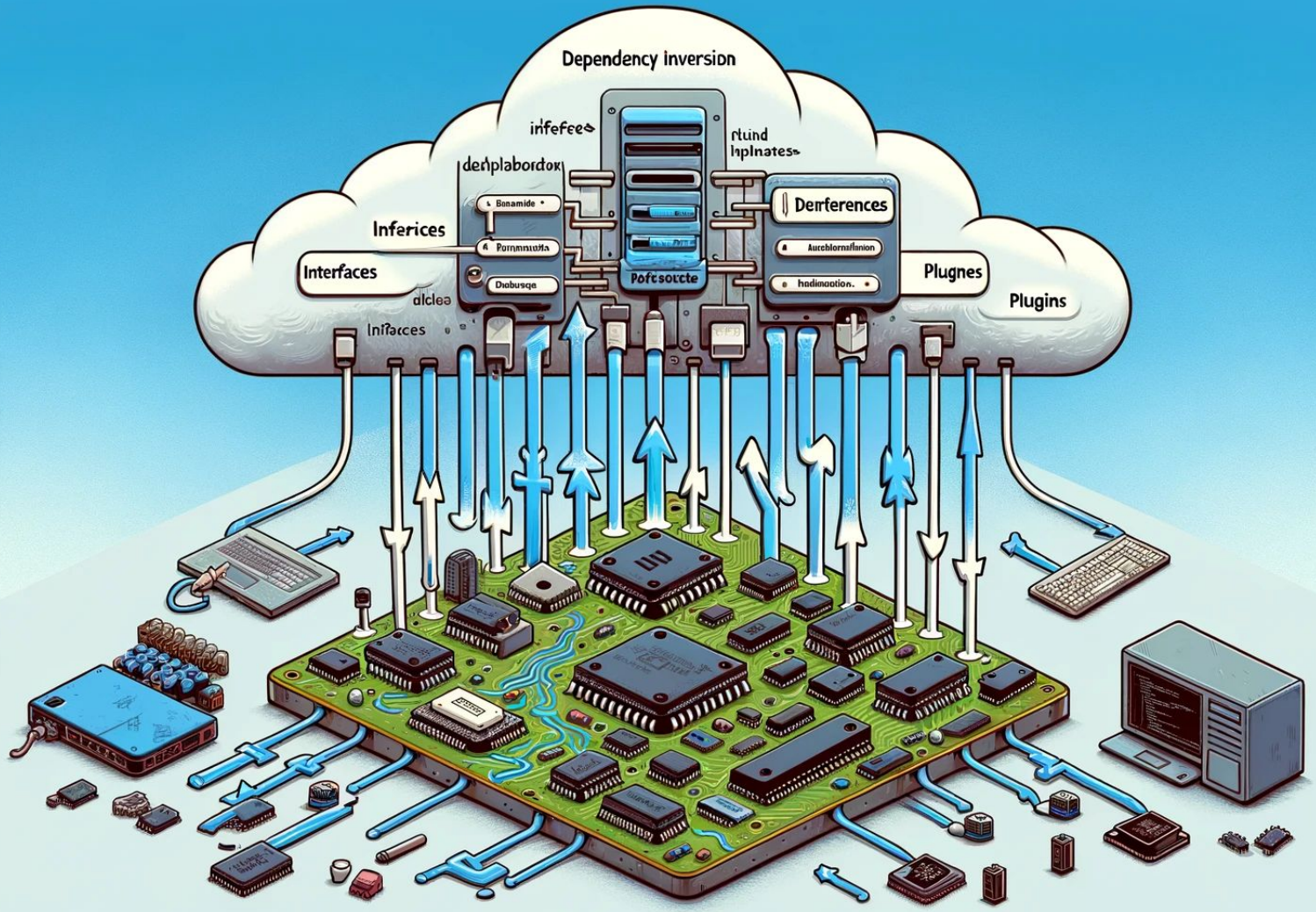
...I PORZĄDKI

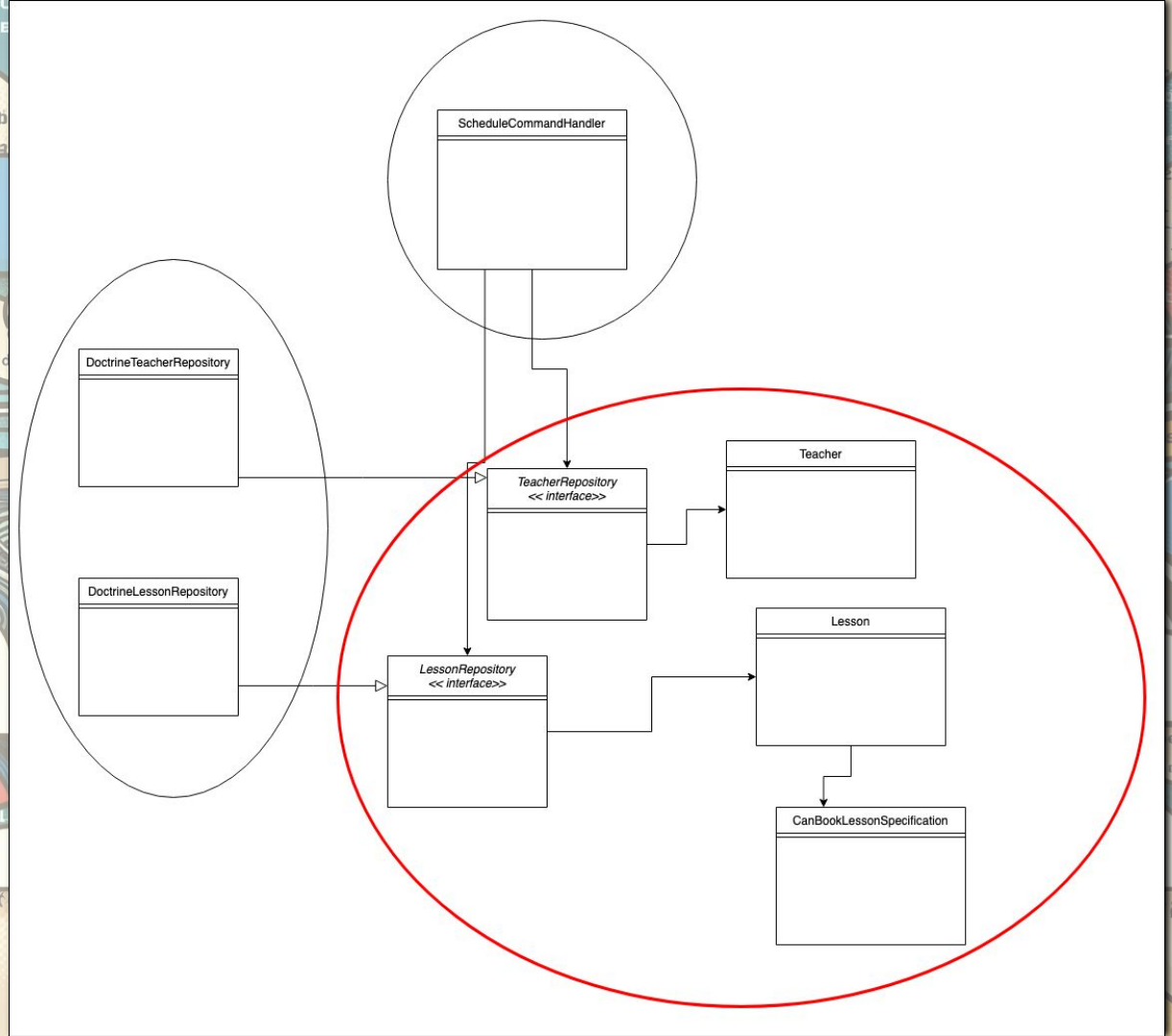
```
class CreateLessonService
{
    public function __construct(
        private CanScheduleTeacherLessonService $canScheduleTeacherLessonService,
        private EntityManagerInterface $entityManager,
        private TeacherRepository $teacherRepository
    ) {
    }
```



```
class CanScheduleTeacherLesson extends Specification
{
    public function __construct(
        private readonly LessonRepository $lessonRepository,
        private readonly \DateTimeImmutable $date
    ) {
    }
```

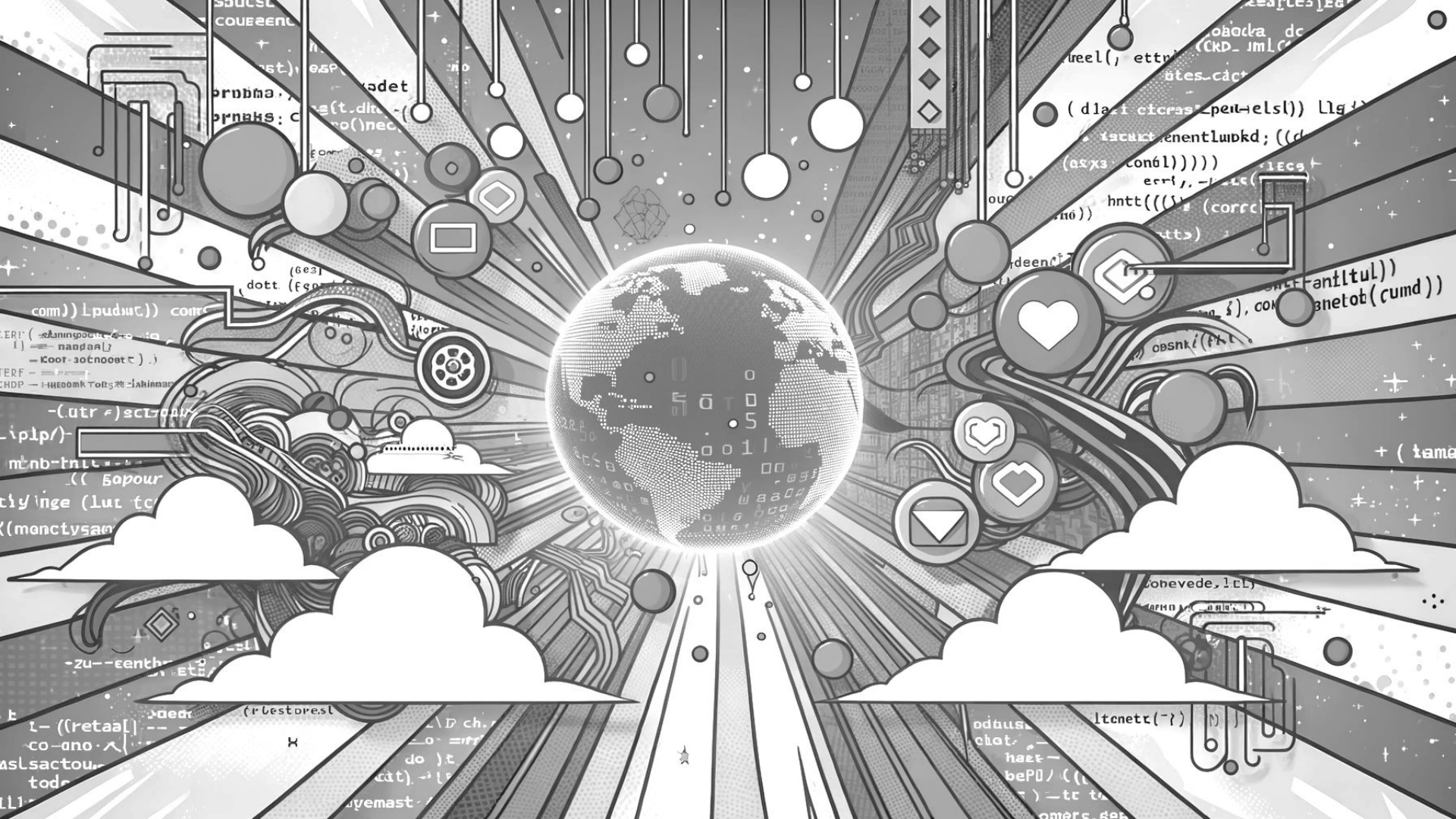
DEPENDENCY INVERSION

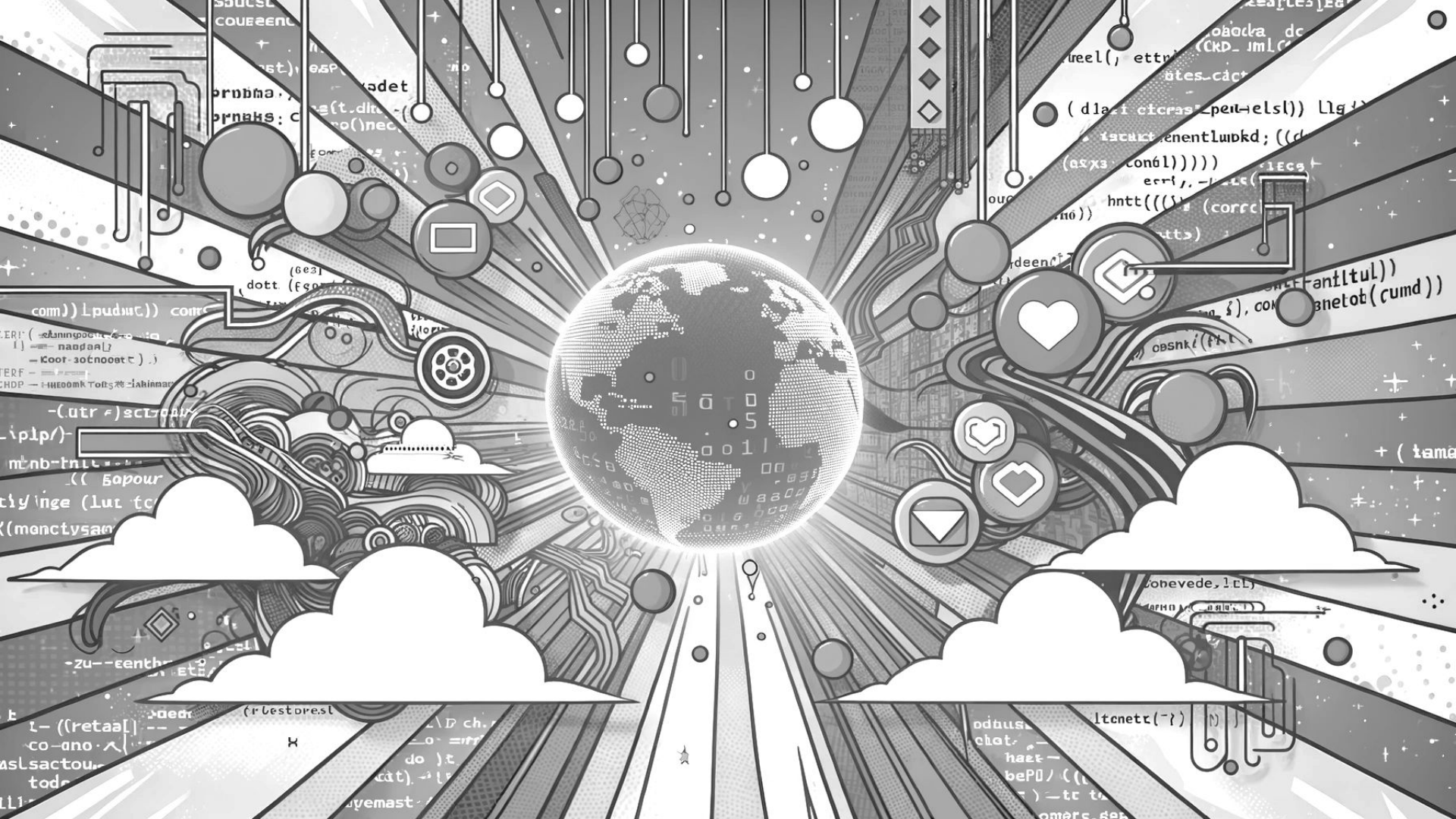


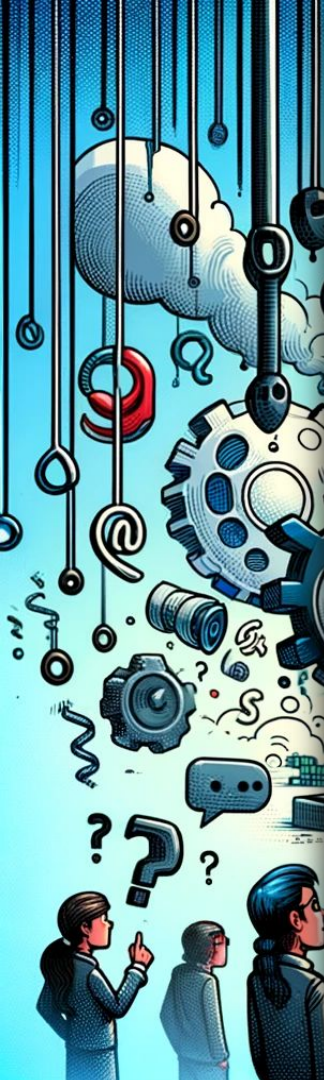

ScheduleCommandHandler
DoctrineTeacherRepository
DoctrineLessonRepository
TeacherRepository
<< interface>>
Teacher
LessonRepository
<< interface>>
Lesson
CanBookLessonSpecification

```php
namespace App\Core\Repository;

interface LessonRepository
{
    public function findByTeacherIdAndDate(string $teacherId, \DateTimeImmutable $date): array;
    public function find(string $id): ?Lesson;
    public function save(Lesson $lesson): void;
}
```

```php
class LessonRepository implements \App\Core\Repository\LessonRepository
{
    public function findByTeacherIdAndDate(
        string $teacherId,
        \DateTimeImmutable $date
    ): array{...}
    public function find(string $id): ?Lesson{...}
    public function save(Lesson $lesson): void{...}
}
```







Q&A

Jakub Ciszak

Starszy programista

BurdaMedia Polska

joindin