

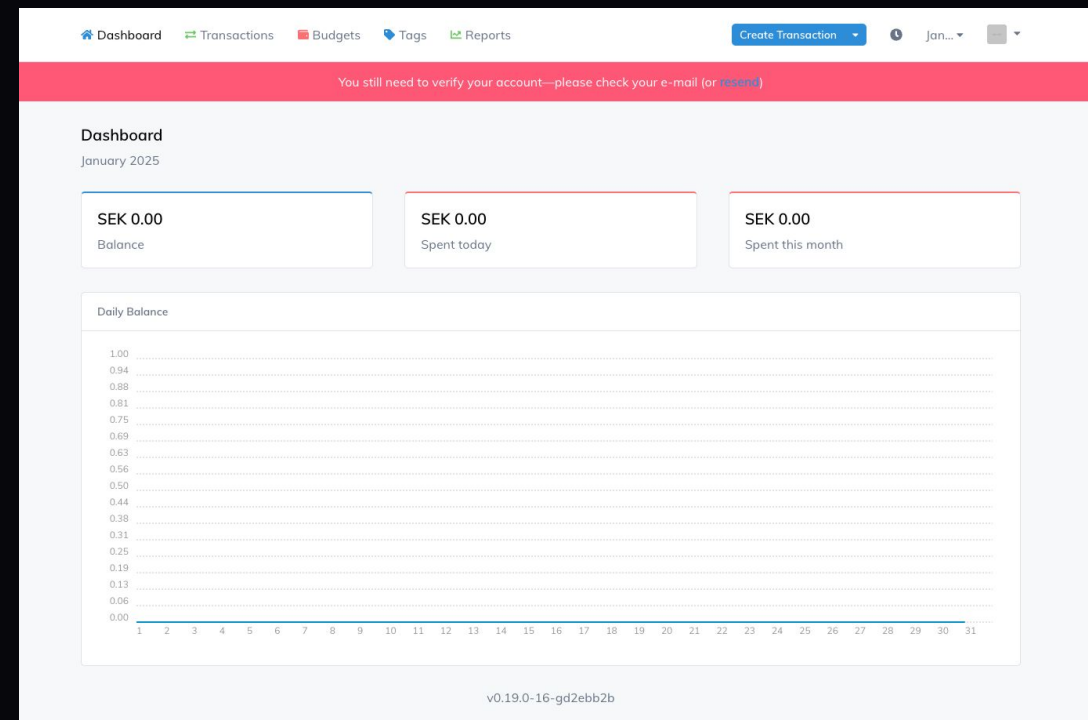
# Budget: Ett PHP-projekt för ekonomihantering

1 Namn  
Budget

2 Ägare  
range-of-motion

3 Språk  
PHP

4 Syfte  
Hantera ekonomi (inkomster, utgifter, budgetar)



# Kravspecifikation

## Funktionella krav

Budget inkluderar användar- och transaktionshantering, stöd för kvitton och CSV-import. Projektet erbjuder också fleranvändarstöd, hantering av olika valutor och språk för en bred användargrupp.

## Tekniska krav

Projektet har Docker-stöd och använder MySQL som databas. Det är värt att notera att det inte finns några explicita kravdokument, vilket ger flexibilitet i utvecklingen men kan också innebära utmaningar för testning och kvalitetssäkring.

2014\_10\_12\_100000\_create\_password\_resets\_table.php

2017\_07\_19\_000000\_create\_tags\_table.php

2017\_07\_20\_000000\_create\_spendings\_table.php

2017\_07\_21\_000000\_create\_earnings\_table.php

2018\_09\_12\_150232\_create\_jobs\_table.php

2018\_09\_13\_112448\_create\_recurrings\_table.php

2018\_10\_02\_185825\_create\_spaces\_table.php

2018\_10\_02\_190502\_create\_user\_space\_table.php

2018\_10\_05\_195046\_add\_overdue\_foreign\_keys.php

2018\_10\_10\_150420\_create\_failed\_jobs\_table.php

2018\_10\_14\_122724\_create\_imports\_table.php

2018\_10\_14\_132003\_add\_import\_id\_column\_to\_spendings\_table.php

2018\_10\_24\_062234\_insert\_major\_western\_currencies.php

2018\_11\_06\_201307\_create\_ideas\_table.php

2018\_11\_19\_180234\_insert\_african\_currencies.php

2018\_11\_23\_161828\_move\_currency\_id\_column\_to\_spaces\_table.php

2018\_12\_17\_174821\_create\_activities\_table.php

2019\_01\_02\_230218\_create\_login\_attempts\_table.php

2020\_06\_05\_153853\_add\_recurring\_id\_column\_to\_earnings\_table.php

2020\_06\_05\_160120\_rename\_type\_column\_to\_interval\_in\_recurrings\_table.php

2020\_06\_05\_160648\_add\_type\_column\_to\_recurrings\_table.php

2020\_06\_13\_151635\_create\_attachments\_table.php

2020\_06\_16\_184356\_insert\_south\_american\_currencies.php

2020\_06\_20\_145102\_insert\_nordic\_currencies.php

2020\_06\_22\_151359\_insert\_remaining\_european\_currencies.php

2020\_06\_28\_202633\_create\_conversion\_rates\_table.php

2020\_06\_30\_204814\_add\_currency\_id\_column\_to\_recurrings\_table.php

2020\_07\_01\_181201\_create\_budgets\_table.php

2020\_07\_01\_230418\_add\_iso\_column\_to\_currencies\_table.php

2020\_07\_15\_195639\_create\_space\_invites\_table.php

2020\_07\_21\_184946\_add\_last\_verification\_mail\_sent\_at\_column\_to\_users\_table.php

2020\_07\_25\_130850\_add\_plan\_column\_to\_users\_table.php

2020\_07\_26\_162432\_add\_stripe\_customer\_id\_column\_to\_users\_table.php

2020\_07\_28\_211805\_create\_widgets\_table.php

2020\_09\_27\_212734\_insert\_turkish\_lira\_into\_currencies\_table.php

2020\_09\_28\_153744\_add\_default\_transaction\_type\_column\_to\_users\_table.php

2020\_09\_28\_181541\_insert\_asian\_currencies.php

2020\_09\_29\_193515\_add\_first\_day\_of\_week\_column\_to\_users\_table.php

# Utvecklingsresa

1

## Start: Oktober 2014

Projektet påbörjades med grundläggande funktioner som användarhantering, taggning och transaktionshantering.

2

## Kontinuerlig utveckling

Över åren har projektet vuxit och förbättrats med nya funktioner och optimeringar.

3

## Senaste ändring: 2024

Stöd för mexikanska pesos lades till, vilket visar på fortsatt utveckling och anpassning till användarnas behov.

4

## Framtid

Planerad uppgradering till Laravel 11 för att hålla projektet uppdaterat och säkert.

# Nuvarande teststrategi



## Enhetstester

Använder PHPUnit för att testa enskilda komponenter och funktioner.



## Funktions-/ Integrationstester

Testar samspelet mellan olika delar av systemet.



## E2E-tester

Använder Laravel Dusk för att simulera användarinteraktioner.



## Bugghantering

Använder GitHub Issues för att spåra och hantera buggar som upptäcks genom tester och manuella kontroller.



# Testning

## Enhetstester

Exempel inkluderar SpaceTest och HelperTest, som fokuserar på att verifiera funktionaliteten hos enskilda komponenter.

## Browser/E2E

SmokeTest via Dusk används för att simulera användarinteraktioner och verifiera systemets övergripande funktionalitet.

## Feature/Integration

CreateEarningTest är ett exempel på hur större funktioner testas i samspel med andra delar av systemet.

## Exploratory testing

Manuell testning ledde till upptäckten av ett 500-fel på /budgets-sidan, vilket visar vikten av olika testmetoder.



# Exempel på Enhetstest

```
public function testAbbreviatedNameAttribute(): void
{
    $space = Space::factory()
        ->create([
            'name' => 'Hello world',
        ]);

    $this->assertEquals('Hel...', $space->abbreviated_name);
}
```

PHP

- Det testar en enda funktion (Single Responsibility)
- Det har ett tydligt förväntat resultat
- Det använder Laravel's factory-system för att skapa testdata

# Exempel på feature test

```
public function testSuccessfulCreation(): void
{
    $response = $this
        ->followingRedirects()
        ->actingAs($this->user)
        ->withSession(['space_id' => $this->space->id])
        ->postJson('/earnings', [
            'date' => date('Y-m-d'),
            'description' => 'Something for the test',
            'amount' => '9.99'
        ]);

    $response
        ->assertStatus(200);
}
```

PHP

- Det testar hela flödet från HTTP-request till respons
- Det simulerar en autentiserad användare med `actingAs()`
- Det testar API-endpoints med JSON-data

# Exempel på browser – End to End test

```
public function testBoot(): void
{
    $this->browse(function (Browser $browser) {
        $browser->visit('/')
                ->assertSee('Log in');
    });
}
```

PHP

Detta är ett så kallat "smoke test" som verifierar att applikationen åtminstone startar och visar inloggningssidan.



# Exempel på mockning

```
public function testSuccessfullySent(): void
{
    $firstVerificationMailSentAt = date('Y-m-d H:i:s', strtotime('7 minute ago'));

    $user = User::factory()->create([
        'verification_token' => 'abc123',
        'last_verification_mail_sent_at' => $firstVerificationMailSentAt
    ]);

    // Förhindrar mailet från att sändas
    Mail::fake();

    (new SendVerificationMailAction())->execute($user->id);

    $this->assertGreaterThan(
        $firstVerificationMailSentAt,
        User::find($user->id)->last_verification_mail_sent_at
    );
}
```

PHP

# Explorativ Testning

**Upptäckter:** 500-fel på /budgets-sidan, inkonsekventa felmeddelanden, problem med filstorlekar, e-postformat och språkhantering.

**Rekommendationer:** Robust felhantering, striktare validering, täckning av edge cases.

# initiala tester och resultat till en början

92

Totalt antal tester

83

Gröna tester

3

Röda tester

157

Assertions

# Resultat efter justering

93

Totalt antal tester

91

Gröna tester

02

Röda tester

164

Assertions

# Initial test coverage

39.88%

Linjetäckning

Indikerar hur stor del av kodbasen som  
nås av tester.

35.35%

Funktions-

Täckning av funktioner och metoder.

22.81%

Klass-

Täckning av klasser och traits.



# Nya tester i BudgetControllerTest

- `user_can_view_budget_index`: Besöker `/budgets`, förväntar sig HTTP 200, verifierar åtkomst till rätt space.
- `user_can_view_create_budget_page`: Besöker `/budgets/create`, förväntar sig HTTP 200, kontrollerar behörighet att skapa budget.
- `user_can_store_new_budget`: Skapar ny budget, sparar data i DB, omdirigerar och kopplar budget till rätt space/tag.

```
class BudgetControllerTest extends TestCase
{
    protected function setUp(): void
    {
        parent::setUp();

        // Skapa en currency först (eftersom space behöver en currency_id)
        $currency = Currency::factory()→create();

        // Skapa space med den skapade currency:n
        $space = Space::factory()→create([
            'currency_id' => $currency→id
        ]);

        session(['space_id' => $space→id]);
    }

    /** @test */
    public function user_can_view_budget_index()
    {
        // Arrange
        $user = User::factory()→create();
        $user→spaces()→attach(session('space_id'));

        // Act
        $response = $this→actingAs($user)
            →get(route('budgets.index'));

        // Assert
        $response→assertStatus(200);
    }
}
```

# Efter tillägg av några tester

40.55%

Linjetäckning

Indikerar hur stor del av kodbasen som  
nås av tester.

36.03%

Funktions-

Täckning av funktioner och metoder.

22.81%

Klass-

Täckning av klasser och traits.

# Styrkor i Testningen

1

Mappstruktur

Lätt att följa och hitta i strukturen

3

**Factories**

Konsekvent hantering av testdata.

2

**Mockning**

Isolerade tester utan beroenden.

# Identifierade Problem

- Låg testtäckning, särskilt för controllers och API
- Bristfällig felhantering i tester
- Problem med icke-unika värden i testdatabasen
- Avsaknad av dokumentation, ingen information om testning i contributors.md



# Sammanfattning

## Status

Budget-projektet visar potential men har för närvarande låg testtäckning. Det finns goda möjligheter till förbättring och vidareutveckling.

## Åtgärder

Prioritera uppgradering till Laravel 11 och stärk den övergripande teststrategin för att öka projektets robusthet och tillförlitlighet.

## Värde

Projektet har en strukturerad kodbas, en solid grund för vidareutveckling och praktiskt Docker-stöd, vilket underlättar deployment och utveckling.

## Nästa steg

Fokusera på att åtgärda identifierade buggar och öka den totala kodtäckningen för att säkerställa projektets långsiktiga hållbarhet och kvalitet.