

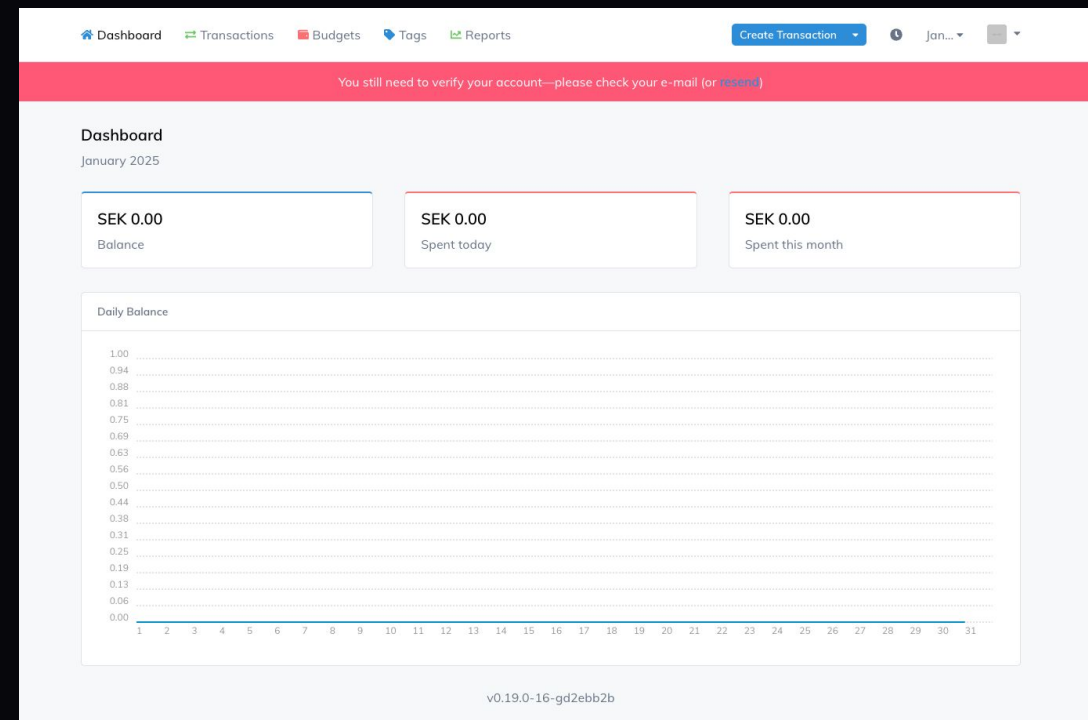
Budget: Ett PHP-projekt för ekonomihantering

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-/Integrations- onstester

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.



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.

Existerande tester och resultat

92

Totalt antal tester

Omfattande testsvit som täcker olika aspekter av systemet.

83

Gröna tester

Majoriteten av testerna passerar, vilket indikerar god grundläggande funktionalitet.

3

Röda tester

Några få tester misslyckas, vilket pekar på områden som behöver åtgärdas.

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

Dataproviders

Effektivt testar olika scenarier.

3

Factories

Konsekvent hantering av testdata.

2

Mockning

Isolerade tester utan beroenden.

4

Policy-tester

Validerar säkerhetsaspekter.

Identifierade Problem

- Låg testtäckning, särskilt för controllers och API
- Bristfällig felhantering i tester
- Problem med icke-unika värden i testdatabasen
- Missade edge cases i validering och säkerhet

Förslag på förbättringar

För att stärka vår testmiljö har vi identifierat flera kritiska förbättringsområden som behöver prioriteras:

1 Saknade tester

Behov av utökade tester för edge cases, felhantering och validering. Särskilt fokus på gränsfall i valutaomvandling, användarinmatning och sessionshantering. Implementation av robusta felhanteringstester för alla kritiska funktioner.

2 Effekt och täckning

Målsättning att öka kodtäckningen till minst 85% och förbättra systemets övergripande tillförlitlighet. Fokus på kritiska affärsprocesser och säkerhetsfunktioner.

3 Dokumentation och underhåll

Förbättrad testdokumentation med tydliga testfall och scenarion. Regelbunden översyn och uppdatering av existerande tester för att säkerställa relevans.

4 Prestandaoptimering

Implementation av systematiska lasttester och prestandamätningar. Etablering av tydliga prestandamål och övervakning av kritiska metrics.

Dessa förbättringar kommer att implementeras stegvis under kommande kvartal, med regelbunden uppföljning och utvärdering av framsteg.



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.