Project Git Styling

Om een nette en georganiseerde repository te onderhouden, stel ik de volgende richtlijnen voor:

Branching Strategie

1. **Master Branch:**
    - Voordat aanpassingen aan de master branch kunnen worden doorgevoerd, moeten deze grondig getest zijn. Dit gebeurt op de `test` branch.
    - De `test` branch wordt gemerged met de `dev` branch wanneer er veranderingen zijn die bedoeld zijn voor de master branch.
2. **Development Branch:**
    - Om aanpassingen aan de `dev` branch te maken, dient er eerst een aparte feature branch te worden aangemaakt.
    - Deze feature branch wordt aan het einde gemerged met de `dev` branch, nadat de wijzigingen zijn goedgekeurd door andere teamleden.

Commit Conventies

We gebruiken het conventionele commit styling format voor onze commit messages. Dit zorgt voor een consistente en duidelijke commit historie.

Format:

```
<type>: <beschrijving>

[optionele inhoud]

[optioneel voetstuk]
```

Type:

- **feat:** Een nieuwe functionaliteit
- **fix:** Een bug fix
- **docs:** Alleen documentatie veranderingen
- **style:** Veranderingen die geen invloed hebben op de betekenis van de code (bijv. white-space, formatting)
- **refactor:** Een code verandering die geen bug oplost, en ook geen nieuwe feature toevoegt
- **perf:** Een code verandering die de performance verbetert
- **test:** Toevoegen van ontbrekende tests of corrigeren van bestaande tests
- **build:** Veranderingen die het build systeem of externe dependencies beïnvloeden
- **ci:** Veranderingen aan onze CI configuratie bestanden en scripts
- **chore:** Andere veranderingen die geen invloed hebben op de src of test bestanden
- **revert:** Terugdraaien van een eerdere commit

Inhoud:

- De inhoud is optioneel en bevat een meer gedetailleerde uitleg over wat er in deze commit is veranderd.
- De inhoud moet in volledige zinnen geschreven worden.

Voetstuk:

- In het voetstuk kunnen referenties naar issues of belangrijke notities worden opgenomen.

Voorbeelden van Commit Messages

Voorbeeld 1

```
feat: added SmsService for creating and sending SMS messages

Introduced a new class called SmsService for handling the creation and submission of
SMS messages using the Vonage API. This service abstracts the API interactions and
simplifies SMS communication within the application.

Additionally, added a custom exception class (SmsSubmissionException) for handling
failed SMS submissions.

Updated the Gradle build configuration to include the Vonage API dependency.
```

Voorbeeld 2

```
fix: corrected the calculation error in InvoiceService

Resolved a bug in the InvoiceService class where the total amount was being calculated
incorrectly due to a rounding issue. The issue was fixed by using BigDecimal for all
monetary calculations.

Added unit tests to cover the rounding scenarios to prevent future regressions.
```

Voorbeeld 3

```
docs: updated README with setup instructions for new developers

Enhanced the README file with detailed setup instructions for new developers joining
the project. This includes environment setup, necessary dependencies, and running the
application locally.

Also added a section on the project's coding conventions and commit message
guidelines.
```

Deze richtlijnen zorgen voor een duidelijke structuur binnen onze Git repository, waardoor samenwerken efficiënter en overzichtelijker wordt.