UNIVERSIDADE ABERTA
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Diagramas de Componentes Git Activity Provider
Padrões de Estrutura

Hugo Gonçalves

Mestrado em Engenharia Informática e Tecnologia Web

2024

Índice

Lista de Figuras

1 Introdução

O *Git Activity Provider* utiliza o padrão de estrutura *Adapter* em dois serviços distintos, o *GitService* e o *AnalyticsService*. De acordo com Gamma et al., 1994, o padrão de estrutura *Adapter*, converte interface de uma classe noutra interface que os clientes conhecem, permitindo que as classes sejam interoperáveis. O primeiro, adapta a interface pública do Github, combinando diversos *endpoints* num modelo de dados único no AP, enquanto o segundo serviço, adapta o modelo de dados do AP e oferece a interface que a Inven!RA espera como definido por Morgado et al., 2023, tal demonstra o diagrama de componentes representado na Figura 1.

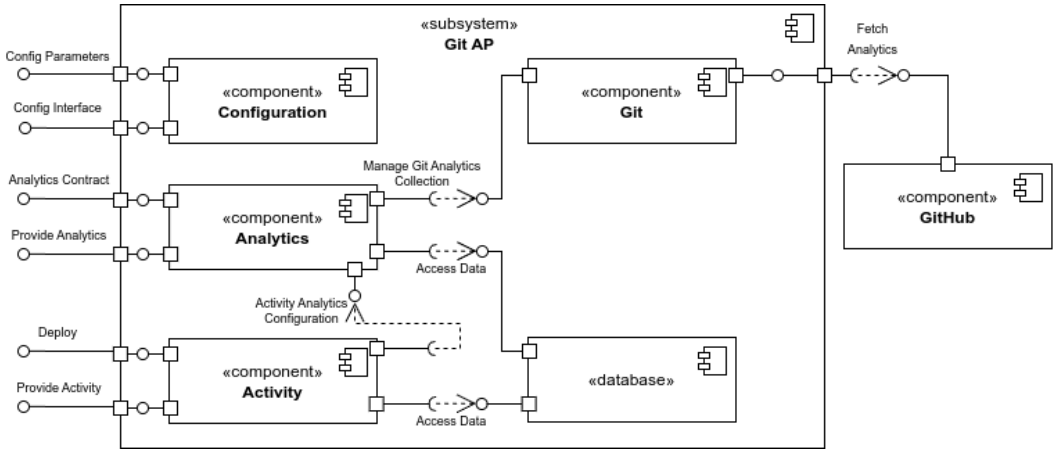

Figura 1: Diagrama de Componentes

2 Casos de uso

Tipicamente, os repositórios centralizados de *git* oferecem uma API para aceder aos dados de repositórios. Estas APIs contém diversos *endpoints* que as *GitRepositoryStrategy* (Gonçalves, 2024) utilizam para coletar as métricas oferecidas pelo *Git Activity Provider*, seguindo assim a interface da API destes.

2.1 Git Service

O *Git Service* age como um *Adapter* que adapta a interface das *GitRepositoryStrategy*, a uma interface que o *Analytics Service* conhece, criando, assim, baixo acoplamento entre o serviço

de *Analytics* e os repositórios *git*. Esta adaptação, está representada a vermelho na chamada própria *1. Create unique data model* no diagrama de sequência da Figura 2.

2.2 Analytics Service

O *Analytics Service* age, também, como um *Adapter*, ao transformar a interface do *Git Service* na interface esperada pela Inven!RA. Além de adaptar a interface, processa também as métricas recolhidas e persiste-as na base de dados já no formato final a enviar para a Inven!RA. Este processamento acontece no caminho de escrita, como representado a vermelho na chamada própria *2. Adapt GitAnalytics* no diagrama de sequência da Figura 2, otimizando assim a latência da leitura. O processamento dos dados no caminho de escrita é abordado por Kleppmann, 2017, Chapter 12, Observing Derived State. Esta otimização é vantajosa para os pedidos descritos pelo diagrama de sequência apresentado na Figura 3 , onde obter métricas é uma simples *query* à base de dados, uma vez que o *Analytics Service*, persistiu os dados no formato em que serão lidos e retornados à Inven!RA.

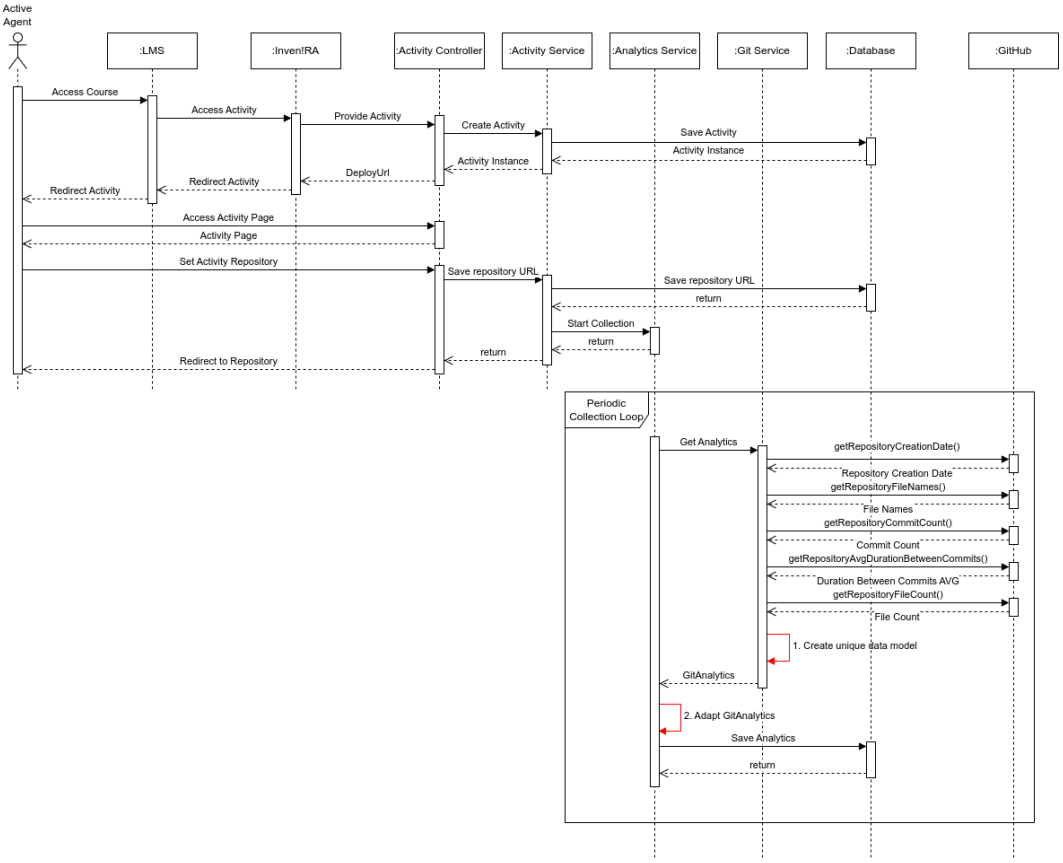

Figura 2: Diagrama de Sequência - Inicio de coleta de métricas

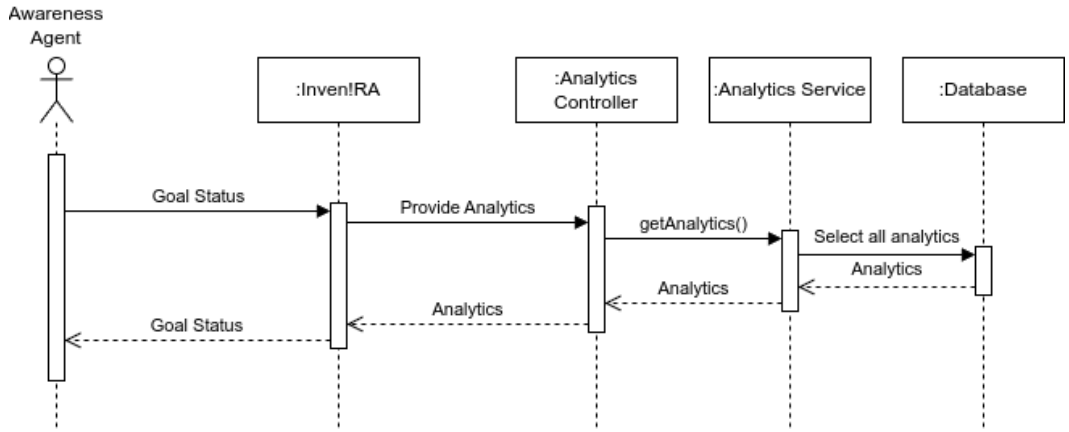

Figura 3: Diagrama de Sequência - Obter métricas

Referências

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994, outubro). *Design Patterns*. Pearson Education.

Gonçalves, H. (2024). GitRepositoryStrategy. *GitHub*. Obtido dezembro 21, 2024, de https://github.com/2100562/GitActivityProvider/blob/main/src/main/java/pt/uab/meiw/aps/git/strategies/GitRepositoryStrategy.java

Kleppmann, M. (2017). *Designing Data-Intensive Applications* (1st). "O'Reilly Media, Inc."

Morgado, L., Coelho, A., Beck, D., Gütl, C., Cassola, F., Baptista, R., van Zeller, M., Pedrosa, D., Cruzeiro, T., Cota, D., Grilo, R., & Schlemmer, E. (2023). Inven!RA Architecture for Sustainable Deployment of Immersive Learning Environments. *Sustainability*, *15*(1). https://doi.org/10.3390/su15010857