**Ронський Святослав Ігорович**

Державний університет “Житомирська політехніка”, м. Житомир

ORCID: 0000-0002-5411-4113

# **Використання AsyncLocalStorage для мультитенантності в NodeJS додатку**

За результатами дослідження популярності мов програмування, що проводилося спільнотою DOU (dou.ua), в загальному рейтингу за 2022 рік мова програмування JavaScript посіла перше місце (18,8%), а мова TypeScript – шосте (10,4%). Якщо розглядати популярність мов в контексті бекенд розробки, то TypeScript посів сьоме місце (3,3%), JavaScript – восьме (3%) [2]. Оскільки ці мови дозволяють розробляти як бекенд так і фронтенд частини додатку, зростає популярність цих мов і кількість написаних на них додатків. Основним середовищем для виконання бекенд коду на цих мовах є платформа NodeJS [3].

Класичні корпоративні клієнт-серверні додатки часто адмініструються в межах однієї організації. Інфраструктура таких додатків зазвичай розрахована на те, що один екземпляр програмного забезпечення працює з одним замовником або організацією. З метою оптимізації інфраструктурних витрат, виникає потреба в можливості роботи декількох замовників з одним додатком. Також, з ростом можливостей вебсерверів та браузерних технологій, все більше розробників вибирають модель розповсюдження “software as a service”. Властивість програмного забезпечення обслуговувати декілька відокремлених замовників, так званих **тенантів**, прозоро розподіляючи наявні ресурси називають **мультитенатністю** [1].

Розглянемо один з варіантів вирішення питання передачі ідентифікатора поточного тенанту між архітектурними шарами у NodeJS додатку. Для прикладу будемо використовувати фреймворк NestJS [4], виключно HTTP контекст та елементи чистої архітектури [5]. З точки зору архітектури, шар бізнес логіки повинен бути незалежним від контексту виклику чи шару персистентності:



В спрощеному вигляді схожу схему залежності шарів буде мати і багатошарова архітектура з використанням принципу інверсії залежностей. Розглянемо можливий вигляд контроллера [4], який буде виступати в ролі Application Layer:

Оскільки наш додаток є мультитенатним, код будь-якого репозиторію повинен знати ідентифікатор поточного тенанту. З іншого боку, оскільки дані та бізнес логіка ізольовані між тенантами, прошарок бізнес логіки може не залежати від тената як такого, тому передача параметру в цей шарє надмірним. Одним з вирішенням проблем може бути створення сервісів в *“request scope”* [4], але такий спосіб не є ефективним з точки використання ресурсів.

Розглянемо вирішення цієї проблеми з використанням *AsyncLocalStorage* [3]. *AsyncLocalStorage* дозволяє зберігати та передавати певні дані через контекст асинхронних викликів. Ми можемо створити інтерцептор [4] для контексту HTTP викликів та підключати його глобально чи окремо для кожного контролера. Інтерцептор може оперувати екземпляром *AsyncLocalStorage* та сетити в нього *tenantId*. В інфраструктурному шарі ми знову можемо працювати з *AsyncLocalStorage* та отримувати з нього *tenantId.* Таким чином ми позбуваємося потреби в передачі *tenantId* через шар бізнес логіки.

**Література**

1. Defining Multi-Tenancy: A Systematic Mapping Study on the Academic and the Industrial Perspective [Електронний ресурс] / J.Kabbedijk, C. Bezemer, S. Jansen, A. Zaidman // Journal of Systems and Software 100 - October 2014. – 2014. – Режим доступу до ресурсу: https://www.researchgate.net/publication/267455810.
2. Шевченко Р. Рейтинг мов програмування 2022 [Електронний ресурс] / Р. Шевченко, І. Яновський. – 2022. – Режим доступу до ресурсу: https://dou.ua/lenta/articles/language-rating-2022.
3. NodeJS documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/docs.
4. NestJS documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.nestjs.com.
5. Мартін Р. Чиста архітектура: Мистецтво розроблення програмного забезпечення / Роберт Мартін. – Харків: Видавництво "Ранок", 2020. – 368 с.