API

▼ 1번 API (특정 지역에 가게 추가하기)

- RegionService.class

```
@Service
@RequiredArgsConstructor
public class RegionService {
    private RegionRepository regionRepository;
    private StoreRepository storeRepository;
    private MissionRepository missionRepository;


        //지역 생성하는 메서드
    @Transactional
    public Region createRegion(String regionName) {
        Region region = new Region();
        region.setName(regionName);
        return regionRepository.save(region);
    }

    // 특정 지역에 가게를 추가하는 메서드
    @Transactional
    public Store addStoreToRegion(Long regionId, String
        Region region = regionRepository.findById(regio
                .orElseThrow(() -> new RuntimeException

        Store store = new Store();
        store.setName(storeName);
        store.setAddress(storeAddress);
        store.setScore(storeScore);
        store.setRegion(region);

        return storeRepository.save(store);
    }
```

- API - RegionController.class

```
@RestController
@RequestMapping("/api/regions")
@Validated
public class RegionController {

    @Autowired
    private RegionService regionService;

    @PostMapping
    public Region createRegion(@Valid @RequestParam Str
        return regionService.createRegion(name);
    }

    @PostMapping("/{regionId}/stores")
    public Store addStoreToRegion(@PathVariable Long re
                                  @Valid @RequestParam
                                  @RequestParam String
                                  @RequestParam Long sc
        return regionService.addStoreToRegion(regionId,
    }
}
```

▼ 2번 API (가게에 리뷰 추가하기)

- ReviewService

```
@Service
public class ReviewService {

    @Autowired
    private StoreRepository storeRepository;

    @Autowired
    private ReviewRepository reviewRepository;

    @Autowired
    private MemberRepository memberRepository;
```

```
    @Transactional
    public Review addReviewToStore(Long storeId, Long m
        Store store = storeRepository.findById(storeId)
                .orElseThrow(() -> new RuntimeException

        Member member = memberRepository.findById(membe
                .orElseThrow(() -> new RuntimeException

        Review review = new Review();
        review.setBody(body);
        review.setScore(score);
        review.setStore(store);
        review.setMember(member);

        return reviewRepository.save(review);
    }
}
```

- ReviewController

```
@RestController
@RequestMapping("/api/reviews")
public class ReviewController {

    @Autowired
    private ReviewService reviewService;

    @PostMapping
    public Review addReviewToStore(@RequestParam Long s
                                   @RequestParam Long m
                                   @Valid @RequestParam
                                   @Valid @RequestParam
        return reviewService.addReviewToStore(storeId, r
    }
}
```

▼ 3번 API (가게에 미션 추가하기)

- MissionService

```
@Service
public class MissionService {
    @Autowired
    private StoreRepository storeRepository;

    @Autowired
    private MissionRepository missionRepository;

    @Transactional
    public Mission addMissionToStore(Long storeId, @Val
        Store store = storeRepository.findById(storeId)
                .orElseThrow(() -> new RuntimeException

        Mission mission = new Mission();
        mission.setReward(reward);
        mission.setDeadline(deadline);
        mission.setMissionSpec(missionSpec);
        mission.setStore(store);

        return missionRepository.save(mission);
    }
}
```

- MissionController

```
@RestController
@RequestMapping("/api/missions")
public class MissionController {

    @Autowired
    private MissionService missionService;

    @PostMapping
    public Mission addMissionToStore(@RequestParam Long
                                     @Valid @RequestPar
                                     @Valid @RequestPar
                                     @Valid @RequestPar
```

```
        LocalDateTime parsedDeadline = LocalDateTime.pa
        return missionService.addMissionToStore(storeId
    }
}
```

▼ 4번 API (가게에 미션을 도전중인 미션에 추가)

- MemberMissionService

```
@Service
public class MemberMissionService {
    @Autowired
    private MemberRepository memberRepository;

    @Autowired
    private MissionRepository missionRepository;

    @Autowired
    private MemberMissionRepository memberMissionReposi

    @Transactional
    public MemberMission addMissionToMember(Long member
        Member member = memberRepository.findById(membe
                .orElseThrow(() -> new RuntimeException

        Mission mission = missionRepository.findById(mi
                .orElseThrow(() -> new RuntimeException

        MemberMission memberMission = new MemberMission
        memberMission.setMember(member);
        memberMission.setMission(mission);
        memberMission.setStatus(status);

        return memberMissionRepository.save(memberMissi
    }
}
```

- MemberMissionController

```
@RestController
@RequestMapping("/api/member-missions")
public class MemberMissionController {

    @Autowired
    private MemberMissionService memberMissionService;

    @PostMapping
    public MemberMission addMissionToMember(@RequestPar
                                            @RequestPar
                                            @Valid @Req
        return memberMissionService.addMissionToMember(
    }
}
```