66. Resolver - post.ts / Mutation - vote()

#graphql #resolver #authentication #mutation #backend #typeorm

Update the mutation for voting

- We update the mutation so that user can change his vote on a post
- We change the method of creating SQL queries so that typeorm will create a transaction using the transaction manager (tm)

/resolvers/post.ts

```
@Mutation(() => Boolean)
  @UseMiddleware(isAuth)
  async vote(
    @Arg("value", () => Int) value: number,
    @Arg("postId", () => Int) postId: number,
    @Ctx() { req }: MyContext
  ) {
    const isUpdoot = value !== -1;
    const realValue = isUpdoot ? 1 : -1;
    const userId = req.session.userId;

    // check to see if the user has voted before
    const updoot = await Updoot.findOne({ where: { postId, userId } });

    // user has voted before and is changing the vote
    if (updoot && updoot.value !== realValue) {
      await getConnection().transaction(async (tm) => {
        // update the updoot table
        await tm.query(
          `
            update updoot
            set value = $1
            where "postId" = $2 and "userId" = $3
          `,
          [realValue, postId, userId]
        );

        // update the post
        await tm.query(
          `
            update post
            set points = points + $1
            where id = $2
          `,
          [2 * realValue, postId] // 2*realValue so that 1 changes to -1 and vice
versa
        );
      });
    } // use has not voted before
    else if (!updoot) {
      await getConnection().transaction(async (tm) => {
        // update the updoot table
        await tm.query(
          `
            insert into updoot("userId", "postId", "value")
            values ($1, $2, $3)
          `,
          [userId, postId, realValue]
        );

        // update the post
        await tm.query(
          `
            update post
            set points = points + $1
            where id = $2
          `,
          [realValue, postId]
        );
      });
    }

    return true;
  }
```