# Лабораторная работа №1

## Анализ кода и выявление проблем

### Введение

В этом задании я проанализировал исходный код файла ajax.php который помогает клиенту получить актуальные данные о рабочих днях, задачах и активности за выбранный период, с учётом ролей и разрешений внутри битрикса. Искал, что в нём не так с архитектурой и качеством кода. Выделил 6 основных проблем, которые сильно влияют на удобство поддержки и развитие. Для каждой проблемы привёл причины, к чему это может привести, и примеры из кода. В конце дал свои идеи, как это исправить.

### 1. Проблема: В коде слишком много всего смешано вместе

Тип: Нарушение принципа единственной ответственности (SRP).

Почему это плохо:

Если в одной функции или файле сразу делают и с датами работать, и с API битрикса, и ошибки обрабатывают то очень тяжело разбираться, что где, и вносить изменения без ошибок.

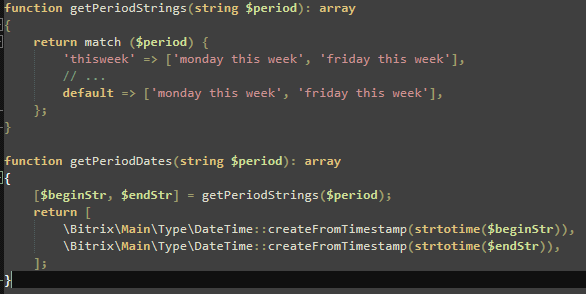
Почему так получилось:

Автор не разделил логику на части, всё написал в одном месте.

Пример:

Как исправить:

Разделить функции на те, что только вычисляют даты, и те, что работают с Битриксом, чтобы легче было менять и тестировать.



### 2. Проблема: Везде “магические” строки вместо констант

Тип: Магические константы.

Почему это плохо:

Если опечататься в названии строки, код сломается, и найти ошибку сложно. Кроме того, сложно быстро найти все места, где используется конкретное значение.

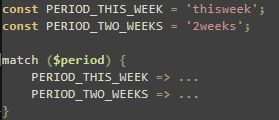
Почему так получилось:

Строки типа "thisweek" и "2weeks" просто прописаны в коде.

Пример:

Как исправить:

Вынести все эти строки в константы с понятными именами, например PERIOD\_THIS\_WEEK и использовать их.



### 3. Проблема: Нет проверки того, что пришло от пользователя

Тип: Отсутствие валидации.

Почему это плохо:

Если пользователь прислал неправильное значение, функция может упасть или работать неправильно.

Почему так получилось:

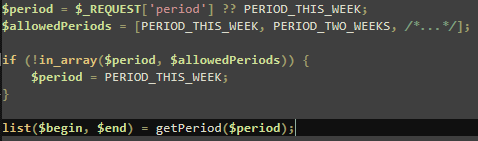
Просто взяли $\_REQUEST['period'] и сразу использовали.

Пример:

$period = $\_REQUEST['period'];  
list($begin, $end) = getPeriod($period);

Как исправить:

Проверять, что параметр есть и соответствует ожидаемым значениям, и если нет — подставлять что-то по умолчанию или возвращать ошибку:



### 4. Проблема: Код жёстко завязан на Bitrix API

Тип: Сильная связанность.

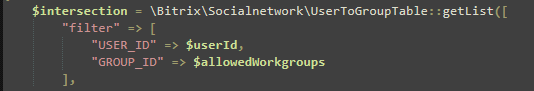
Почему это плохо:

Если надо протестировать код отдельно или перенести на другой проект — будет сложно.

Почему так получилось:

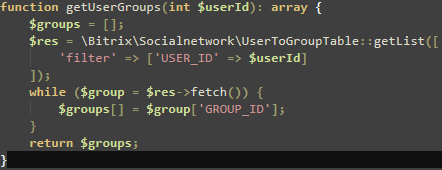
Везде прямые вызовы \Bitrix\Socialnetwork\UserToGroupTable::getList() и т.п., без обёрток.

Пример:



Как исправить:

Создать отдельные функции, которые делают эти запросы, и вызывать их — так проще менять и тестировать.



### 5. Проблема: Нет комментариев и документации

Тип: Отсутствие документации.

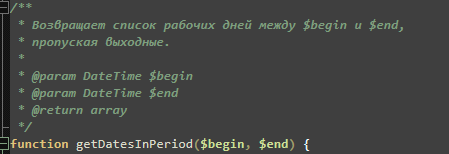
Почему это плохо:

Через пару месяцев или другому человеку будет сложно понять, зачем что-то сделано и как использовать функции.

Почему так получилось:

Просто забыли добавить комменты.

Пример:



Функция getDatesInPeriod сложная, но без единого пояснения.

Как исправить:

Добавить короткие комментарии или PHPDoc, объяснить, что делает каждая функция, какие параметры и что возвращает.

### 6. Проблема: Жёстко прописаны выходные дни

Тип: Негибкая логика.

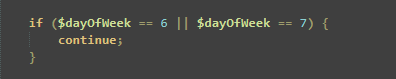
Почему это плохо:

Если появятся праздники или изменится правило учёта выходных — менять код будет неудобно.

Почему так получилось:

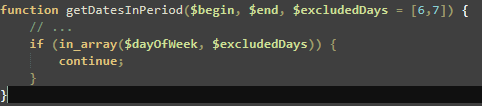
Просто написано пропускать субботу и воскресенье, без возможности настройки.

Пример:



Как исправить:

Сделать, чтобы список выходных передавался в функцию и можно было его менять.



## Итог

Провёл разбор — код работает, но сделать его удобнее, надежнее и гибче очень просто, если:  
Разделить логику на части, использовать константы вместо строк, проверять данные от пользователя, создать обёртки для Bitrix, писать комментарии, сделать правила учёта выходных гибкими.