PHP Backend Developer recruitment task

Create a RESTful API

Purpose of the API is to allow storing, listing and updating football match predictions.
There are 2 kinds of predictions:

- 3 way result (1 - Home team win, X - Draw, 2 - Away team win)
- correct score (e.g. 3:2, 1:0, 5:3)

The API must provides 3 endpoints:

- POST /v1/predictions (create a prediction)
- GET /v1/predictions (list all predictions)
- PUT /v1/predictions/:id/status (update status of the prediction)

Prediction object is a set of the following properties:

Property name	**Type**	**Description**
id	int unsigned	auto-generated ID
event_id	int unsigned	any unsigned int number
market_type	enum(1x2, correct_score)	type of the prediction
prediction	string	value of the prediction, e.g. 1, 2, X for 1x2, 3:2, 5:1 and so on for correct_score
status	enum(win, lost, unresolved)	unresolved by default
created_at	timestamp	automatically set by the API
updated_at	timestamp	automatically set by the API

Example prediction objects

Example 1

```
{
  "id": 1,
  "event_id": 123,
  "market_type": "1x2",
  "prediction": "X",
  "status": "unresolved"
}
```

Example 2

```
{
  "id": 2,
  "event_id": 532,
  "market_type": "correct_score",
  "prediction": "3:2",
  "status": "win"
}
```

Endpoints

POST /v1/predictions

- gets application/json content type with the following request body (example)
    - {
    -   "event_id": 1,
    -   "market_type": "correct_score",
    -   "prediction": "3:2"
    - }
- validates event_id, market_type and prediction fields, returns 400 bad request response if fails
- id, status, created_at and updated_at should be set automatically
- once the valid prediction was posted, stores it and returns 204 response

GET /v1/predictions

- returns application/json collection with all the predictions
- doesn’t provide pagination and filtering

PUT /v1/predictions/:id/status

- allows to set status of the given prediction
- gets application/json content type with the following request body (example)
    - {
    -   "status": "lost"
    - }
- validates whether the given prediction exists, return 404 response if not
- validates status value, returns 400 bad request response if fails
- updates status of the prediction
- returns 204 on success

Requirements

1. PHP framework: any
2. database layer: any
3. authentication: no

Additional points

1. Separate database layer with Service-Repository pattern
2. Unit tests
3. Postman collection
4. .gitignore
5. Logger (e.g. Monolog or other compliant with PSR-3)
6. PSR-2
7. Vagrant or Docker setup