Приложение: <https://github.com/deflorator1980/Mosaic>

Запуск: mvn spring-boot:run

На адрес localhost:8080/response POST-запросом предедаются данные о фигурах, размещённых пользователем. Где “figure” - номер фигуры, “vert” и “hor” - вертикальные и горизонтальные координаты полей, занимаемых фигурой, счёт с нуля, с левого верхнего угла.



Например:

“

{

"tetrominos": [

{

"figure": "4",

"xy": [

{

"vert": 0,

"hor": 0

},

{

"vert": 0,

"hor": 1

},

{

"vert": 1,

"hor": 0

}

]

},

{

"figure": "7",

"xy": [

{

"vert": 2,

"hor": 4

},

{

"vert": 2,

"hor": 5

},

{

"vert": 3,

"hor": 4

}

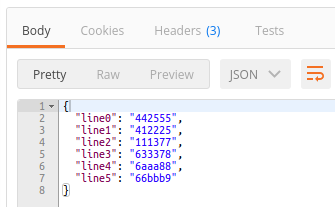
]

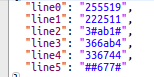
}

]

}

“  
В результате возвращается поле 6х6 с соответствующими номерами фигур:



Если остаются пустые поля, они обозначаются знаком “#”

В качестве JSON-клиента использовал Postman (<https://www.getpostman.com/>)  
  
Про алгоритм.  
 Если я правильно понял, заполнение тетрамино – это NP-сложная задача т.е. идеально решение гарантивано только полным перебором конфиграций. Теоретическое обоснование <https://arxiv.org/abs/cs/0210020>.   
У нас 36 клеток, 9 фигур, округлённо в 4-х позициях т.е. (36\*4) в 9 степени – это примерно 10 в 19 степени.   
Современные настольные компьютеры выдают 0.1 терафлопс (10 в 11-й операций в секунду) т.е. им потребуется 10 в 8-й секунд т.е. около 3 лет.

Суперкомпьютеры выдают петафлопсы (10 в 15-й оп/сек) – могут решить минут за 15.   
  
Всё это оценки по максимуму – многие комбианации при переборе будут невалидны изначально – реальная размерность порядка на 2-3 меньше ожидается.   
Но есть примерно неделя.   
  
Поэтому алгоритм выбрал простой: берутся сначала крупные фигуры, потом более мелкие. Изначально фигуры развёрнуты в позициях, представленных в идеальном решении, потом пробуют другие варианты разворота.   
  
  
Проверка вводимых пользователем данных не осуществляется, так как проверять лучше на юае(UI) (на каком-нибудь Ангуляре).