An exception occurred in driver: SQLSTATE[HY000] [2002] Connection timed out

Exceptions 3

Doctrine\\DBAL\\Exception
ConnectionException

in vendor/doctrine/dbal/lib/Doctrine/DBAL/Driver/**AbstractMySQLDriver.php** (line 112)

```
107.             case \'1227\':
108.             case \'1370\':
109.             case \'1429\':
110.             case \'2002\':
111.             case \'2005\':
112.                 return new ConnectionException($message, $exception);
113.
114.             case \'2006\':
115.                 return new ConnectionLost($message, $exception);
116.
117.             case \'1048\':
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**DBALException.php** -> **convertException** (line 182)

```
177.         if ($driverEx instanceof DriverException) {
178.             return $driverEx;
179.         }
180.
181.         if ($driver instanceof ExceptionConverterDriver && $driverEx instanceof Depre
182.             return $driver->convertException($msg, $driverEx);
183.         }
184.
185.         return new Exception($msg, 0, $driverEx);
186.     }
187.
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**DBALException.php** :: **wrapException** (line 169)

```
164.      *
165.      * @return Exception
```

```
166.      */
167.     public static function driverException(Driver $driver, Throwable $driverEx)
168.     {
169.         return self::wrapException($driver, $driverEx, \'An exception occurred in dri
170.     }
171.
172.     /**
173.      * @return Exception
174.      */
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/Driver/PDOMySql/**Driver.php** :: **driverException** (line 31)

```
26.                 $username,
27.                 $password,
28.                 $driverOptions
29.             );
30.         } catch (PDOException $e) {
31.             throw Exception::driverException($this, $e);
32.         }
33.
34.         return $conn;
35.     }
36.
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**Connection.php** -> **connect** (line 412)

```
407.
408.         $driverOptions = $this->params[\'driverOptions\'] ?? [];
409.         $user          = $this->params[\'user\'] ?? null;
410.         $password      = $this->params[\'password\'] ?? null;
411.
412.         $this->_conn = $this->_driver->connect($this->params, $user, $password, $driv
413.
414.         $this->transactionNestingLevel = 0;
415.
416.         if ($this->autoCommit === false) {
417.             $this->beginTransaction();
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**Connection.php** -> **connect** (line 1952)

```
1947.      *
1948.      * @return DriverConnection
```

```
1949.      */
1950.     public function getWrappedConnection()
1951.     {
1952.         $this->connect();
1953.
1954.         assert($this->_conn !== null);
1955.
1956.         return $this->_conn;
1957.     }
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**Connection.php** -> **getWrappedConnection** (line 980)

```
975.      * @param mixed                $value
976.      * @param int|string|Type|null $type
977.      */
978.     public function quote($value, $type = ParameterType::STRING)
979.     {
980.         $connection = $this->getWrappedConnection();
981.
982.         [$value, $bindingType] = $this->getBindingInfo($value, $type);
983.
984.         return $connection->quote($value, $bindingType);
985.     }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **quote** (line 467)

```
462.             if ($class->discriminatorValue !== null) { // discriminators can be 0
463.                 $values[] = $conn->quote($class->discriminatorValue);
464.             }
465.
466.             foreach ($class->subClasses as $subclassName) {
467.                 $values[] = $conn->quote($this->em->getClassMetadata($subclassName)->
468.             }
469.
470.             $sqlTableAlias = ($this->useSqlTableAliases)
471.                 ? $this->getSQLTableAlias($class->getTableName(), $dqlAlias) . \'.\'
472.                 : \'\';
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **_generateDiscriminatorColumnConditionSQL** (line 987)

```
982.
983.                     $conditions[] = $sourceTableAlias . \'.\' . $quotedTargetColumn .
```

```
984.                 }
985.
986.                 // Apply remaining inheritance restrictions
987.                 $discrSql = $this->_generateDiscriminatorColumnConditionSQL([$joinedD
988.
989.                 if ($discrSql) {
990.                     $conditions[] = $discrSql;
991.                 }
992.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkJoinAssociationDeclaration** (line 1196)

```
1191.                 }
1192.
1193.                 break;
1194.
1195.             case ($joinDeclaration instanceof AST\\JoinAssociationDeclaration):
1196.                 $sql .= $this->walkJoinAssociationDeclaration($joinDeclaration, $join
1197.                 break;
1198.         }
1199.
1200.         return $sql;
1201.     }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkJoin** (line 852)

```
847.         if ($identificationVariableDecl->indexBy) {
848.             $this->walkIndexBy($identificationVariableDecl->indexBy);
849.         }
850.
851.         foreach ($identificationVariableDecl->joins as $join) {
852.             $sql .= $this->walkJoin($join);
853.         }
854.
855.         return $sql;
856.     }
857.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkIdentificationVariableDeclaration** (line 830)

```
825.     {
826.         $identificationVarDecls = $fromClause->identificationVariableDeclarations;
```

```
827.         $sqlParts = [];
828.
829.         foreach ($identificationVarDecls as $identificationVariableDecl) {
830.             $sqlParts[] = $this->walkIdentificationVariableDeclaration($identificatic
831.         }
832.
833.         return \' FROM \' . implode(\', \', $sqlParts);
834.     }
835.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkFromClause** (line 535)

```
530.     {
531.         $limit    = $this->query->getMaxResults();
532.         $offset   = $this->query->getFirstResult();
533.         $lockMode = $this->query->getHint(Query::HINT_LOCK_MODE);
534.         $sql      = $this->walkSelectClause($AST->selectClause)
535.             . $this->walkFromClause($AST->fromClause)
536.             . $this->walkWhereClause($AST->whereClause);
537.
538.         if ($AST->groupByClause) {
539.             $sql .= $this->walkGroupByClause($AST->groupByClause);
540.         }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Tools/Pagination/**CountOutputWalker.php** -> **walkSelectStatement** (line 92)

```
87.     {
88.         if ($this->platform->getName() === "mssql") {
89.             $AST->orderByClause = null;
90.         }
91.
92.         $sql = parent::walkSelectStatement($AST);
93.
94.         if ($AST->groupByClause) {
95.             return sprintf(
96.                 \'SELECT %s AS dctrn_count FROM (%s) dctrn_table\',
97.                 $this->platform->getCountExpression(\'*\'),
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/Exec/**SingleSelectExecutor.php** -> **walkSelectStatement** (line 42)

```
37.      * @param \\Doctrine\\ORM\\Query\\AST\\SelectStatement $AST
38.      * @param \\Doctrine\\ORM\\Query\\SqlWalker           $sqlWalker
```

```
39.      */
40.     public function __construct(SelectStatement $AST, SqlWalker $sqlWalker)
41.     {
42.         $this->_sqlStatements = $sqlWalker->walkSelectStatement($AST);
43.     }
44.
45.     /**
46.      * {@inheritDoc}
47.      */
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **__construct** (line 288)

```
283.                 return ($primaryClass->isInheritanceTypeJoined())
284.                     ? new Exec\\MultiTableUpdateExecutor($AST, $this)
285.                     : new Exec\\SingleTableDeleteUpdateExecutor($AST, $this);
286.
287.             default:
288.                 return new Exec\\SingleSelectExecutor($AST, $this);
289.         }
290.     }
291.
292.     /**
293.      * Generates a unique, short SQL table alias.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**Parser.php** -> **getExecutor** (line 411)

```
406.
407.         $outputWalkerClass = $this->customOutputWalker ?: SqlWalker::class;
408.         $outputWalker      = new $outputWalkerClass($this->query, $this->parserResult
409.
410.         // Assign an SQL executor to the parser result
411.         $this->parserResult->setSqlExecutor($outputWalker->getExecutor($AST));
412.
413.         return $this->parserResult;
414.     }
415.
416.     /**
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**Query.php** -> **parse** (line 287)

```
282.         }
283.
284.         // Cache miss.
```

```
285.         $parser = new Parser($this);
286.
287.         $this->_parserResult = $parser->parse();
288.
289.         $queryCache->save($hash, $this->_parserResult, $this->_queryCacheTTL);
290.
291.         return $this->_parserResult;
292.     }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**Query.php** -> **_parse** (line 299)

```
294.     /**
295.      * {@inheritdoc}
296.      */
297.     protected function _doExecute()
298.     {
299.         $executor = $this->_parse()->getSqlExecutor();
300.
301.         if ($this->_queryCacheProfile) {
302.             $executor->setQueryCacheProfile($this->_queryCacheProfile);
303.         } else {
304.             $executor->removeQueryCacheProfile();
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**AbstractQuery.php** -> **_doExecute** (line 1000)

```
995.
996.                 $cache->save($cacheKey, $result, $queryCacheProfile->getLifetime());
997.             };
998.         }
999.
1000.         $stmt = $this->_doExecute();
1001.
1002.         if (is_numeric($stmt)) {
1003.             $setCacheEntry($stmt);
1004.
1005.             return $stmt;
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**AbstractQuery.php** -> **executeIgnoreQueryCache** (line 954)

```
949.     {
950.         if ($this->cacheable && $this->isCacheEnabled()) {
951.             return $this->executeUsingQueryCache($parameters, $hydrationMode);
952.         }
```

```
953.
954.         return $this->executeIgnoreQueryCache($parameters, $hydrationMode);
955.     }
956.
957.     /**
958.      * Execute query ignoring second level cache.
959.      *
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**AbstractQuery.php** -> **execute** (line 781)

```
776.      *
777.      * @return array
778.      */
779.     public function getScalarResult()
780.     {
781.         return $this->execute(null, self::HYDRATE_SCALAR);
782.     }
783.
784.     /**
785.      * Get exactly one result or null.
786.      *
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Tools/Pagination/**Paginator.php** -> **getScalarResult** (line 126)

```
121.      */
122.     public function count()
123.     {
124.         if ($this->count === null) {
125.             try {
126.                 $this->count = (int) array_sum(array_map(\'current\', $this->getCount
127.             } catch (NoResultException $e) {
128.                 $this->count = 0;
129.             }
130.         }
131.
```

Paginator -> **count** ()
in src/Repository/**NsdPrdplWorkitemsRepository.php** (line 297)

```
292.
293.         $query = $qb->getQuery();
294.
295.         $paginator = new Paginator($query, $fetchJoinCollection = true);
```

```
296.
297.         $count = \\count($paginator);
298.
299.         if (null === $perPage) {
300.             return new PartialList($count, $paginator->getQuery()->getResult());
301.         }
302.
```

NsdPrdplWorkitemsRepository -> **getWorkitemsByFilter** (*object*(WorkitemFilter), *array*(\'wi.date\' => \'ASC\', \'s.subjectName\' => \'ASC\', \'wi.id\' => \'ASC\'), 1, 100)
in src/RestController/**WorkitemsController.php** (line 233)

```
228.          */
229.         $workitemList = $this->workitemsRepository->getWorkitemsByFilter(
230.             $workitemFilter,
231.             [\'wi.date\' => \'ASC\', \'s.subjectName\' => \'ASC\', \'wi.id\' => \'ASC
232.             (int) $paramFetcher->get(\'page\', true),
233.             (int) $paramFetcher->get(\'perPage\', true)
234.         );
235.
236.         $result = \\array_map(
237.             function (NsdPrdplWorkitem $workitem): array {
238.                 $wiArray = $workitem->toArray();
```

in vendor/symfony/http-kernel/**HttpKernel.php** -> **getWorkitemsAction** (line 158)

```
153.         $this->dispatcher->dispatch($event, KernelEvents::CONTROLLER_ARGUMENTS);
154.         $controller = $event->getController();
155.         $arguments = $event->getArguments();
156.
157.         // call controller
158.         $response = $controller(...$arguments);
159.
160.         // view
161.         if (!$response instanceof Response) {
162.             $event = new ViewEvent($this, $request, $type, $response);
163.             $this->dispatcher->dispatch($event, KernelEvents::VIEW);
```

in vendor/symfony/http-kernel/**HttpKernel.php** -> **handleRaw** (line 80)

```
75.     public function handle(Request $request, $type = HttpKernelInterface::MASTER_REQU
76.     {
77.         $request->headers->set(\'X-Php-Ob-Level\', (string) ob_get_level());
```

```
78.
79.         try {
80.             return $this->handleRaw($request, $type);
81.         } catch (\\Exception $e) {
82.             if ($e instanceof RequestExceptionInterface) {
83.                 $e = new BadRequestHttpException($e->getMessage(), $e);
84.             }
85.             if (false === $catch) {
```

in vendor/symfony/http-kernel/**Kernel.php** -> **handle** (line 201)

```
196.         $this->boot();
197.         ++$this->requestStackSize;
198.         $this->resetServices = true;
199.
200.         try {
201.             return $this->getHttpKernel()->handle($request, $type, $catch);
202.         } finally {
203.             --$this->requestStackSize;
204.         }
205.     }
206.
```

Kernel -> **handle** (*object*(Request))
in public/**index.php** (line 52)

```
47.  $kernel = new Kernel($environment, (bool) $debug);
48.
49.  // When using the HttpCache, you need to call the method in your front controller ins
50.  //Request::enableHttpMethodParameterOverride();
51.  $request = Request::createFromGlobals();
52.  $response = $kernel->handle($request);
53.  $response->send();
54.  $kernel->terminate($request, $response);
55.
```

Doctrine\\DBAL\\Driver\\PDO

Exception

SQLSTATE[HY000] [2002] Connection timed out

in vendor/doctrine/dbal/lib/Doctrine/DBAL/Driver/PDO/**Exception.php** (line 18)

```
13.  */
14. final class Exception extends PDOException
15. {
16.     public static function new(\\PDOException $exception): self
17.     {
18.         return new self($exception);
19.     }
20. }
21.
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/Driver/**PDOConnection.php** :: **new** (line 44)

```
39.         try {
40.             parent::__construct($dsn, (string) $user, (string) $password, (array) $op
41.             $this->setAttribute(PDO::ATTR_STATEMENT_CLASS, [Statement::class, []]);
42.             $this->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
43.         } catch (PDOException $exception) {
44.             throw Exception::new($exception);
45.         }
46.     }
47.
48.     /**
49.      * {@inheritdoc}
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/Driver/PDOMySql/**Driver.php** -> **__construct** (line 25)

```
20.      */
21.     public function connect(array $params, $username = null, $password = null, array
22.     {
23.         try {
24.             $conn = new PDO\\Connection(
25.                 $this->constructPdoDsn($params),
26.                 $username,
27.                 $password,
28.                 $driverOptions
29.             );
30.         } catch (PDOException $e) {
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**Connection.php** -> **connect** (line 412)

```
407.
408.         $driverOptions = $this->params[\'driverOptions\'] ?? [];
```

```
409.         $user           = $this->params[\'user\'] ?? null;
410.         $password       = $this->params[\'password\'] ?? null;
411.
412.         $this->_conn = $this->_driver->connect($this->params, $user, $password, $driv
413.
414.         $this->transactionNestingLevel = 0;
415.
416.         if ($this->autoCommit === false) {
417.             $this->beginTransaction();
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**Connection.php** -> **connect** (line 1952)

```
1947.      *
1948.      * @return DriverConnection
1949.      */
1950.     public function getWrappedConnection()
1951.     {
1952.         $this->connect();
1953.
1954.         assert($this->_conn !== null);
1955.
1956.         return $this->_conn;
1957.     }
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**Connection.php** -> **getWrappedConnection** (line 980)

```
975.      * @param mixed                $value
976.      * @param int|string|Type|null $type
977.      */
978.     public function quote($value, $type = ParameterType::STRING)
979.     {
980.         $connection = $this->getWrappedConnection();
981.
982.         [$value, $bindingType] = $this->getBindingInfo($value, $type);
983.
984.         return $connection->quote($value, $bindingType);
985.     }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **quote** (line 467)

```
462.             if ($class->discriminatorValue !== null) { // discriminators can be 0
463.                 $values[] = $conn->quote($class->discriminatorValue);
464.             }
```

```
465.
466.             foreach ($class->subClasses as $subclassName) {
467.                 $values[] = $conn->quote($this->em->getClassMetadata($subclassName)->
468.             }
469.
470.             $sqlTableAlias = ($this->useSqlTableAliases)
471.                 ? $this->getSQLTableAlias($class->getTableName(), $dqlAlias) . \'.\'
472.                 : \'\';
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **_generateDiscriminatorColumnConditionSQL** (line 987)

```
982.
983.                     $conditions[] = $sourceTableAlias . \'.\' . $quotedTargetColumn .
984.                 }
985.
986.                 // Apply remaining inheritance restrictions
987.                 $discrSql = $this->_generateDiscriminatorColumnConditionSQL([$joinedD
988.
989.                 if ($discrSql) {
990.                     $conditions[] = $discrSql;
991.                 }
992.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkJoinAssociationDeclaration** (line 1196)

```
1191.                 }
1192.
1193.                 break;
1194.
1195.             case ($joinDeclaration instanceof AST\\JoinAssociationDeclaration):
1196.                 $sql .= $this->walkJoinAssociationDeclaration($joinDeclaration, $join
1197.                 break;
1198.         }
1199.
1200.         return $sql;
1201.     }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkJoin** (line 852)

```
847.         if ($identificationVariableDecl->indexBy) {
848.             $this->walkIndexBy($identificationVariableDecl->indexBy);
```

```
849.         }
850.
851.         foreach ($identificationVariableDecl->joins as $join) {
852.             $sql .= $this->walkJoin($join);
853.         }
854.
855.         return $sql;
856.     }
857.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkIdentificationVariableDeclaration** (line 830)

```
825.     {
826.         $identificationVarDecls = $fromClause->identificationVariableDeclarations;
827.         $sqlParts = [];
828.
829.         foreach ($identificationVarDecls as $identificationVariableDecl) {
830.             $sqlParts[] = $this->walkIdentificationVariableDeclaration($identificatic
831.         }
832.
833.         return \' FROM \' . implode(\', \', $sqlParts);
834.     }
835.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkFromClause** (line 535)

```
530.     {
531.         $limit    = $this->query->getMaxResults();
532.         $offset   = $this->query->getFirstResult();
533.         $lockMode = $this->query->getHint(Query::HINT_LOCK_MODE);
534.         $sql      = $this->walkSelectClause($AST->selectClause)
535.             . $this->walkFromClause($AST->fromClause)
536.             . $this->walkWhereClause($AST->whereClause);
537.
538.         if ($AST->groupByClause) {
539.             $sql .= $this->walkGroupByClause($AST->groupByClause);
540.         }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Tools/Pagination/**CountOutputWalker.php** -> **walkSelectStatement** (line 92)

```
87.     {
88.         if ($this->platform->getName() === "mssql") {
```

```
89.             $AST->orderByClause = null;
90.         }
91.
92.         $sql = parent::walkSelectStatement($AST);
93.
94.         if ($AST->groupByClause) {
95.             return sprintf(
96.                 \'SELECT %s AS dctrn_count FROM (%s) dctrn_table\',
97.                 $this->platform->getCountExpression(\'*\'),
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/Exec/**SingleSelectExecutor.php** -> **walkSelectStatement** (line 42)

```
37.      * @param \\Doctrine\\ORM\\Query\\AST\\SelectStatement $AST
38.      * @param \\Doctrine\\ORM\\Query\\SqlWalker           $sqlWalker
39.      */
40.     public function __construct(SelectStatement $AST, SqlWalker $sqlWalker)
41.     {
42.         $this->_sqlStatements = $sqlWalker->walkSelectStatement($AST);
43.     }
44.
45.     /**
46.      * {@inheritDoc}
47.      */
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **__construct** (line 288)

```
283.                 return ($primaryClass->isInheritanceTypeJoined())
284.                     ? new Exec\\MultiTableUpdateExecutor($AST, $this)
285.                     : new Exec\\SingleTableDeleteUpdateExecutor($AST, $this);
286.
287.             default:
288.                 return new Exec\\SingleSelectExecutor($AST, $this);
289.         }
290.     }
291.
292.     /**
293.      * Generates a unique, short SQL table alias.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**Parser.php** -> **getExecutor** (line 411)

```
406.
407.         $outputWalkerClass = $this->customOutputWalker ?: SqlWalker::class;
408.         $outputWalker      = new $outputWalkerClass($this->query, $this->parserResult
```

```
409.
410.         // Assign an SQL executor to the parser result
411.         $this->parserResult->setSqlExecutor($outputWalker->getExecutor($AST));
412.
413.         return $this->parserResult;
414.     }
415.
416.     /**
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**Query.php** -> **parse** (line 287)

```
282.         }
283.
284.         // Cache miss.
285.         $parser = new Parser($this);
286.
287.         $this->_parserResult = $parser->parse();
288.
289.         $queryCache->save($hash, $this->_parserResult, $this->_queryCacheTTL);
290.
291.         return $this->_parserResult;
292.     }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**Query.php** -> **_parse** (line 299)

```
294.     /**
295.      * {@inheritdoc}
296.      */
297.     protected function _doExecute()
298.     {
299.         $executor = $this->_parse()->getSqlExecutor();
300.
301.         if ($this->_queryCacheProfile) {
302.             $executor->setQueryCacheProfile($this->_queryCacheProfile);
303.         } else {
304.             $executor->removeQueryCacheProfile();
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**AbstractQuery.php** -> **_doExecute** (line 1000)

```
995.
996.                 $cache->save($cacheKey, $result, $queryCacheProfile->getLifetime());
997.             };
998.         }
```

```
999.
1000.         $stmt = $this->_doExecute();
1001.
1002.         if (is_numeric($stmt)) {
1003.             $setCacheEntry($stmt);
1004.
1005.             return $stmt;
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**AbstractQuery.php** -> **executeIgnoreQueryCache** (line 954)

```
949.     {
950.         if ($this->cacheable && $this->isCacheEnabled()) {
951.             return $this->executeUsingQueryCache($parameters, $hydrationMode);
952.         }
953.
954.         return $this->executeIgnoreQueryCache($parameters, $hydrationMode);
955.     }
956.
957.     /**
958.      * Execute query ignoring second level cache.
959.      *
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**AbstractQuery.php** -> **execute** (line 781)

```
776.      *
777.      * @return array
778.      */
779.     public function getScalarResult()
780.     {
781.         return $this->execute(null, self::HYDRATE_SCALAR);
782.     }
783.
784.     /**
785.      * Get exactly one result or null.
786.      *
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Tools/Pagination/**Paginator.php** -> **getScalarResult** (line 126)

```
121.      */
122.     public function count()
123.     {
124.         if ($this->count === null) {
125.             try {
126.                 $this->count = (int) array_sum(array_map('current', $this->getCount
```

```
127.             } catch (NoResultException $e) {
128.                 $this->count = 0;
129.             }
130.         }
131.
```

Paginator -> **count** ()
in src/Repository/**NsdPrdplWorkitemsRepository.php** (line 297)

```
292.
293.         $query = $qb->getQuery();
294.
295.         $paginator = new Paginator($query, $fetchJoinCollection = true);
296.
297.         $count = \\count($paginator);
298.
299.         if (null === $perPage) {
300.             return new PartialList($count, $paginator->getQuery()->getResult());
301.         }
302.
```

NsdPrdplWorkitemsRepository -> **getWorkitemsByFilter** (*object*(WorkitemFilter), *array*(\'wi.date\' => \'ASC\', \'s.subjectName\' => \'ASC\', \'wi.id\' => \'ASC\'), 1, 100)
in src/RestController/**WorkitemsController.php** (line 233)

```
228.          */
229.         $workitemList = $this->workitemsRepository->getWorkitemsByFilter(
230.             $workitemFilter,
231.             [\'wi.date\' => \'ASC\', \'s.subjectName\' => \'ASC\', \'wi.id\' => \'ASC
232.             (int) $paramFetcher->get(\'page\', true),
233.             (int) $paramFetcher->get(\'perPage\', true)
234.         );
235.
236.         $result = \\array_map(
237.             function (NsdPrdplWorkitem $workitem): array {
238.                 $wiArray = $workitem->toArray();
```

in vendor/symfony/http-kernel/**HttpKernel.php** -> **getWorkitemsAction** (line 158)

```
153.         $this->dispatcher->dispatch($event, KernelEvents::CONTROLLER_ARGUMENTS);
154.         $controller = $event->getController();
155.         $arguments = $event->getArguments();
156.
```

```
157.         // call controller
158.         $response = $controller(...$arguments);
159.
160.         // view
161.         if (!$response instanceof Response) {
162.             $event = new ViewEvent($this, $request, $type, $response);
163.             $this->dispatcher->dispatch($event, KernelEvents::VIEW);
```

in vendor/symfony/http-kernel/**HttpKernel.php** -> **handleRaw** (line 80)

```
75.     public function handle(Request $request, $type = HttpKernelInterface::MASTER_REQU
76.     {
77.         $request->headers->set(\'X-Php-Ob-Level\', (string) ob_get_level());
78.
79.         try {
80.             return $this->handleRaw($request, $type);
81.         } catch (\\Exception $e) {
82.             if ($e instanceof RequestExceptionInterface) {
83.                 $e = new BadRequestHttpException($e->getMessage(), $e);
84.             }
85.             if (false === $catch) {
```

in vendor/symfony/http-kernel/**Kernel.php** -> **handle** (line 201)

```
196.         $this->boot();
197.         ++$this->requestStackSize;
198.         $this->resetServices = true;
199.
200.         try {
201.             return $this->getHttpKernel()->handle($request, $type, $catch);
202.         } finally {
203.             --$this->requestStackSize;
204.         }
205.     }
206.
```

Kernel -> **handle** (*object*(Request))
in public/**index.php** (line 52)

```
47. $kernel = new Kernel($environment, (bool) $debug);
48.
49. // When using the HttpCache, you need to call the method in your front controller ins
50. //Request::enableHttpMethodParameterOverride();
```

```
51. $request = Request::createFromGlobals();
52. $response = $kernel->handle($request);
53. $response->send();
54. $kernel->terminate($request, $response);
55.
```

PDOException

SQLSTATE[HY000] [2002] Connection timed out

in vendor/doctrine/dbal/lib/Doctrine/DBAL/Driver/**PDOConnection.php** (line 40)

```
35.      * @throws PDOException In case of an error.
36.      */
37.     public function __construct($dsn, $user = null, $password = null, ?array $options
38.     {
39.         try {
40.             parent::__construct($dsn, (string) $user, (string) $password, (array) $op
41.             $this->setAttribute(PDO::ATTR_STATEMENT_CLASS, [Statement::class, []]);
42.             $this->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
43.         } catch (PDOException $exception) {
44.             throw Exception::new($exception);
45.         }
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/Driver/**PDOConnection.php** -> **__construct** (line 40)

```
35.      * @throws PDOException In case of an error.
36.      */
37.     public function __construct($dsn, $user = null, $password = null, ?array $options
38.     {
39.         try {
40.             parent::__construct($dsn, (string) $user, (string) $password, (array) $op
41.             $this->setAttribute(PDO::ATTR_STATEMENT_CLASS, [Statement::class, []]);
42.             $this->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
43.         } catch (PDOException $exception) {
44.             throw Exception::new($exception);
45.         }
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/Driver/PDOMySql/**Driver.php** -> **__construct** (line 25)

```
20.      */
21.     public function connect(array $params, $username = null, $password = null, array
```

```
22.     {
23.         try {
24.             $conn = new PDO\\Connection(
25.                 $this->constructPdoDsn($params),
26.                 $username,
27.                 $password,
28.                 $driverOptions
29.             );
30.         } catch (PDOException $e) {
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**Connection.php** -> **connect** (line 412)

```
407.
408.         $driverOptions = $this->params[\'driverOptions\'] ?? [];
409.         $user          = $this->params[\'user\'] ?? null;
410.         $password      = $this->params[\'password\'] ?? null;
411.
412.         $this->_conn = $this->_driver->connect($this->params, $user, $password, $driv
413.
414.         $this->transactionNestingLevel = 0;
415.
416.         if ($this->autoCommit === false) {
417.             $this->beginTransaction();
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**Connection.php** -> **connect** (line 1952)

```
1947.      *
1948.      * @return DriverConnection
1949.      */
1950.     public function getWrappedConnection()
1951.     {
1952.         $this->connect();
1953.
1954.         assert($this->_conn !== null);
1955.
1956.         return $this->_conn;
1957.     }
```

in vendor/doctrine/dbal/lib/Doctrine/DBAL/**Connection.php** -> **getWrappedConnection** (line 980)

```
975.      * @param mixed                $value
976.      * @param int|string|Type|null $type
977.      */
```

```
978.     public function quote($value, $type = ParameterType::STRING)
979.     {
980.         $connection = $this->getWrappedConnection();
981.
982.         [$value, $bindingType] = $this->getBindingInfo($value, $type);
983.
984.         return $connection->quote($value, $bindingType);
985.     }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **quote** (line 467)

```
462.             if ($class->discriminatorValue !== null) { // discriminators can be 0
463.                 $values[] = $conn->quote($class->discriminatorValue);
464.             }
465.
466.             foreach ($class->subClasses as $subclassName) {
467.                 $values[] = $conn->quote($this->em->getClassMetadata($subclassName)->
468.             }
469.
470.             $sqlTableAlias = ($this->useSqlTableAliases)
471.                 ? $this->getSQLTableAlias($class->getTableName(), $dqlAlias) . \'.\'
472.                 : \'\';
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **_generateDiscriminatorColumnConditionSQL** (line 987)

```
982.
983.                     $conditions[] = $sourceTableAlias . \'.\' . $quotedTargetColumn .
984.                 }
985.
986.                 // Apply remaining inheritance restrictions
987.                 $discrSql = $this->_generateDiscriminatorColumnConditionSQL([$joinedC
988.
989.                 if ($discrSql) {
990.                     $conditions[] = $discrSql;
991.                 }
992.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkJoinAssociationDeclaration** (line 1196)

```
1191.                 }
1192.
```

```
1193.                 break;
1194.
1195.             case ($joinDeclaration instanceof AST\\JoinAssociationDeclaration):
1196.                 $sql .= $this->walkJoinAssociationDeclaration($joinDeclaration, $join
1197.                 break;
1198.         }
1199.
1200.         return $sql;
1201.     }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkJoin** (line 852)

```
847.         if ($identificationVariableDecl->indexBy) {
848.             $this->walkIndexBy($identificationVariableDecl->indexBy);
849.         }
850.
851.         foreach ($identificationVariableDecl->joins as $join) {
852.             $sql .= $this->walkJoin($join);
853.         }
854.
855.         return $sql;
856.     }
857.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkIdentificationVariableDeclaration** (line 830)

```
825.     {
826.         $identificationVarDecls = $fromClause->identificationVariableDeclarations;
827.         $sqlParts = [];
828.
829.         foreach ($identificationVarDecls as $identificationVariableDecl) {
830.             $sqlParts[] = $this->walkIdentificationVariableDeclaration($identificatic
831.         }
832.
833.         return \' FROM \' . implode(\', \', $sqlParts);
834.     }
835.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **walkFromClause** (line 535)

```
530.     {
531.         $limit    = $this->query->getMaxResults();
```

```
532.         $offset    = $this->query->getFirstResult();
533.         $lockMode = $this->query->getHint(Query::HINT_LOCK_MODE);
534.         $sql       = $this->walkSelectClause($AST->selectClause)
535.             . $this->walkFromClause($AST->fromClause)
536.             . $this->walkWhereClause($AST->whereClause);
537.
538.         if ($AST->groupByClause) {
539.             $sql .= $this->walkGroupByClause($AST->groupByClause);
540.         }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Tools/Pagination/**CountOutputWalker.php** -> **walkSelectStatement** (line 92)

```
87.     {
88.         if ($this->platform->getName() === "mssql") {
89.             $AST->orderByClause = null;
90.         }
91.
92.         $sql = parent::walkSelectStatement($AST);
93.
94.         if ($AST->groupByClause) {
95.             return sprintf(
96.                 \'SELECT %s AS dctrn_count FROM (%s) dctrn_table\',
97.                 $this->platform->getCountExpression(\'*\'),
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/Exec/**SingleSelectExecutor.php** -> **walkSelectStatement** (line 42)

```
37.      * @param \\Doctrine\\ORM\\Query\\AST\\SelectStatement $AST
38.      * @param \\Doctrine\\ORM\\Query\\SqlWalker           $sqlWalker
39.      */
40.     public function __construct(SelectStatement $AST, SqlWalker $sqlWalker)
41.     {
42.         $this->_sqlStatements = $sqlWalker->walkSelectStatement($AST);
43.     }
44.
45.     /**
46.      * {@inheritDoc}
47.      */
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**SqlWalker.php** -> **__construct** (line 288)

```
283.                 return ($primaryClass->isInheritanceTypeJoined())
284.                     ? new Exec\\MultiTableUpdateExecutor($AST, $this)
```

```
285.                     : new Exec\\SingleTableDeleteUpdateExecutor($AST, $this);
286.
287.             default:
288.                 return new Exec\\SingleSelectExecutor($AST, $this);
289.         }
290.     }
291.
292.     /**
293.      * Generates a unique, short SQL table alias.
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Query/**Parser.php** -> **getExecutor** (line 411)

```
406.
407.         $outputWalkerClass = $this->customOutputWalker ?: SqlWalker::class;
408.         $outputWalker      = new $outputWalkerClass($this->query, $this->parserResult
409.
410.         // Assign an SQL executor to the parser result
411.         $this->parserResult->setSqlExecutor($outputWalker->getExecutor($AST));
412.
413.         return $this->parserResult;
414.     }
415.
416.     /**
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**Query.php** -> **parse** (line 287)

```
282.         }
283.
284.         // Cache miss.
285.         $parser = new Parser($this);
286.
287.         $this->_parserResult = $parser->parse();
288.
289.         $queryCache->save($hash, $this->_parserResult, $this->_queryCacheTTL);
290.
291.         return $this->_parserResult;
292.     }
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**Query.php** -> **_parse** (line 299)

```
294.     /**
295.      * {@inheritdoc}
296.      */
```

```
297.     protected function _doExecute()
298.     {
299.         $executor = $this->_parse()->getSqlExecutor();
300.
301.         if ($this->_queryCacheProfile) {
302.             $executor->setQueryCacheProfile($this->_queryCacheProfile);
303.         } else {
304.             $executor->removeQueryCacheProfile();
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**AbstractQuery.php** -> **_doExecute** (line 1000)

```
995.
996.                 $cache->save($cacheKey, $result, $queryCacheProfile->getLifetime());
997.             };
998.         }
999.
1000.         $stmt = $this->_doExecute();
1001.
1002.         if (is_numeric($stmt)) {
1003.             $setCacheEntry($stmt);
1004.
1005.             return $stmt;
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**AbstractQuery.php** -> **executeIgnoreQueryCache** (line 954)

```
949.     {
950.         if ($this->cacheable && $this->isCacheEnabled()) {
951.             return $this->executeUsingQueryCache($parameters, $hydrationMode);
952.         }
953.
954.         return $this->executeIgnoreQueryCache($parameters, $hydrationMode);
955.     }
956.
957.     /**
958.      * Execute query ignoring second level cache.
959.      *
```

in vendor/doctrine/orm/lib/Doctrine/ORM/**AbstractQuery.php** -> **execute** (line 781)

```
776.      *
777.      * @return array
778.      */
779.     public function getScalarResult()
```

```
780.     {
781.         return $this->execute(null, self::HYDRATE_SCALAR);
782.     }
783.
784.     /**
785.      * Get exactly one result or null.
786.      *
```

in vendor/doctrine/orm/lib/Doctrine/ORM/Tools/Pagination/**Paginator.php** -> **getScalarResult** (line 126)

```
121.      */
122.     public function count()
123.     {
124.         if ($this->count === null) {
125.             try {
126.                 $this->count = (int) array_sum(array_map(\'current\', $this->getCount
127.             } catch (NoResultException $e) {
128.                 $this->count = 0;
129.             }
130.         }
131.
```

Paginator -> **count** ()
in src/Repository/**NsdPrdplWorkitemsRepository.php** (line 297)

```
292.
293.         $query = $qb->getQuery();
294.
295.         $paginator = new Paginator($query, $fetchJoinCollection = true);
296.
297.         $count = \\count($paginator);
298.
299.         if (null === $perPage) {
300.             return new PartialList($count, $paginator->getQuery()->getResult());
301.         }
302.
```

NsdPrdplWorkitemsRepository -> **getWorkitemsByFilter** (*object*(WorkitemFilter), *array*(\'wi.date\' => \'ASC\', \'s.subjectName\' => \'ASC\', \'wi.id\' => \'ASC\'), 1, 100)
in src/RestController/**WorkitemsController.php** (line 233)

```
228.          */
229.         $workitemList = $this->workitemsRepository->getWorkitemsByFilter(
230.             $workitemFilter,
```

```
231.             [\'wi.date\' => \'ASC\', \'s.subjectName\' => \'ASC\', \'wi.id\' => \'ASC
232.             (int) $paramFetcher->get(\'page\', true),
233.             (int) $paramFetcher->get(\'perPage\', true)
234.         );
235.
236.         $result = \\array_map(
237.             function (NsdPrdplWorkitem $workitem): array {
238.                 $wiArray = $workitem->toArray();
```

in vendor/symfony/http-kernel/**HttpKernel.php** -> **getWorkitemsAction** (line 158)

```
153.         $this->dispatcher->dispatch($event, KernelEvents::CONTROLLER_ARGUMENTS);
154.         $controller = $event->getController();
155.         $arguments = $event->getArguments();
156.
157.         // call controller
158.         $response = $controller(...$arguments);
159.
160.         // view
161.         if (!$response instanceof Response) {
162.             $event = new ViewEvent($this, $request, $type, $response);
163.             $this->dispatcher->dispatch($event, KernelEvents::VIEW);
```

in vendor/symfony/http-kernel/**HttpKernel.php** -> **handleRaw** (line 80)

```
75.     public function handle(Request $request, $type = HttpKernelInterface::MASTER_REQU
76.     {
77.         $request->headers->set(\'X-Php-Ob-Level\', (string) ob_get_level());
78.
79.         try {
80.             return $this->handleRaw($request, $type);
81.         } catch (\\Exception $e) {
82.             if ($e instanceof RequestExceptionInterface) {
83.                 $e = new BadRequestHttpException($e->getMessage(), $e);
84.             }
85.             if (false === $catch) {
```

in vendor/symfony/http-kernel/**Kernel.php** -> **handle** (line 201)

```
196.         $this->boot();
197.         ++$this->requestStackSize;
198.         $this->resetServices = true;
199.
```

```
200.         try {
201.             return $this->getHttpKernel()->handle($request, $type, $catch);
202.         } finally {
203.             --$this->requestStackSize;
204.         }
205.     }
206.
```

Kernel -> **handle** (*object*(Request))
in public/**index.php** (line 52)

```
47.  $kernel = new Kernel($environment, (bool) $debug);
48.
49.  // When using the HttpCache, you need to call the method in your front controller ins
50.  //Request::enableHttpMethodParameterOverride();
51.  $request = Request::createFromGlobals();
52.  $response = $kernel->handle($request);
53.  $response->send();
54.  $kernel->terminate($request, $response);
55.
```