

Trabalho Final

Sistemas Operacionais I

Florianópolis, 27 de Maio de 2015.

Tema do projeto: aplicação em PC para configuração de rede, captura de tela, instalação de pacotes do SO e outras ações de um sistema de gerenciamento de aula em laboratório de informática.

Sobre o projeto: o projeto final da disciplina “*Sistemas Operacionais I*” é denominado *LabSpy* e será disponibilizado de forma gratuita e de código-aberto. O objetivo principal do programa é melhorar o processo de ensino-aprendizagem em laboratórios de computadores fornecendo ao docente ferramentas para monitoramento e controle dos computadores dos discentes.

Grupo:

Caique Rodrigues Marques,
Emmanuel Podestá Junior,
Fernando Jorge Mota,
Fernando Paladini.

Conteúdo abordado:

- 1-3 Primeira Entrega
 - 1. Requisitos
 - 1.1. Requisitos funcionais
 - 1.2. Requisitos não-funcionais
 - 2. Diagramas
 - 3. Funcionamento planejado (**importante**)
- 4. Segunda Entrega
 - 4.1. Novidades
 - 4.2. Pré-requisitos
 - 4.3. Comunicação
 - 4.4. Visualizar Tela do Aluno
 - 4.5. Controle Remoto
 - 4.6. Processo para iniciar e usar as funcionalidades
- 5. Diagramas

1-3 Primeira Entrega

1 Requisitos

1.1 Requisitos Funcionais

Os requisitos funcionais do projeto *LabSpy* são os que seguem abaixo:

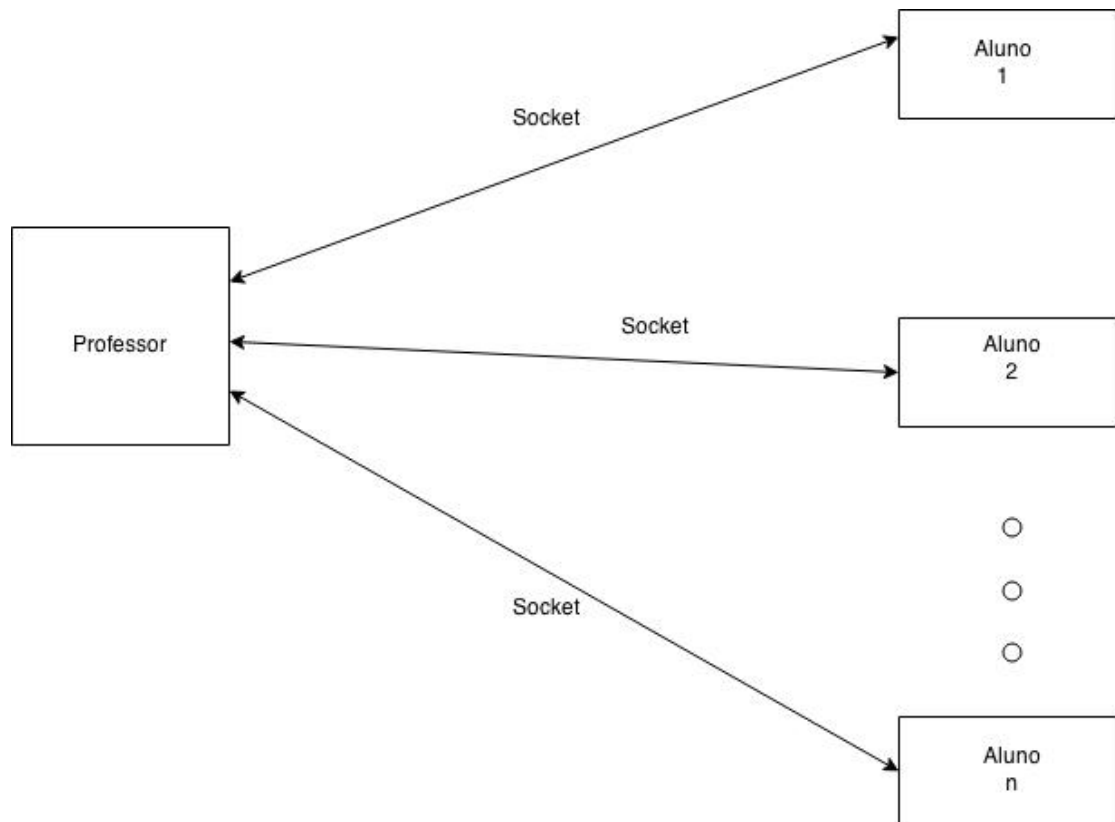
- **Overview das áreas de trabalho:** o programa deve permitir que o docente tenha um *overview* da área de trabalho de todos os computadores conectados na rede, mostrando as imagens dos *desktops* ao vivo ou em uma frequência relativamente alta;
- **Visualização de área de trabalho:** O programa deve permitir que o docente possa ter uma visualização completa (toda a área de tela do programa) da área de trabalho de um computador em específico;
- **Controle remoto do mouse:** O programa deve permitir que o docente controle o *mouse* do computador de um discente a qualquer momento;
- **Controle remoto do teclado:** O programa deve permitir que o docente controle o *teclado* do computador de um discente a qualquer momento;
- **Bloqueio de computadores:** O programa deve permitir que o docente bloqueie e desbloqueie um computador da rede em qualquer dado momento;
- **Proxy definido pelo professor:** O programa deve permitir que o docente possa bloquear sites de serem acessados pelos alunos em qualquer dado momento.
- **Chat:** O programa deve permitir que o docente possa abrir um chat e enviar mensagens instantâneas para todas os estudantes (computadores) da turma.

1.2 Requisitos Não-Funcionais

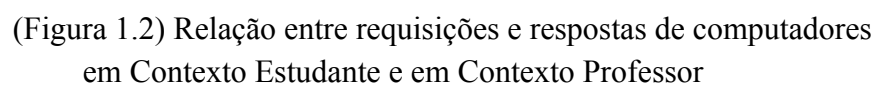
Os requisitos não-funcionais do projeto são os que seguem abaixo:

- O projeto deve ser escrito principalmente na linguagem de programação Java;
- O projeto deve ser disponibilizado de forma gratuita;
- O projeto deve ser *open-source* (código-aberto);
- O projeto deve ser multiplataforma e executar sem maiores problemas em ambientes Linux (principalmente a distribuição Ubuntu) e em ambientes Windows (Windows XP e superiores).

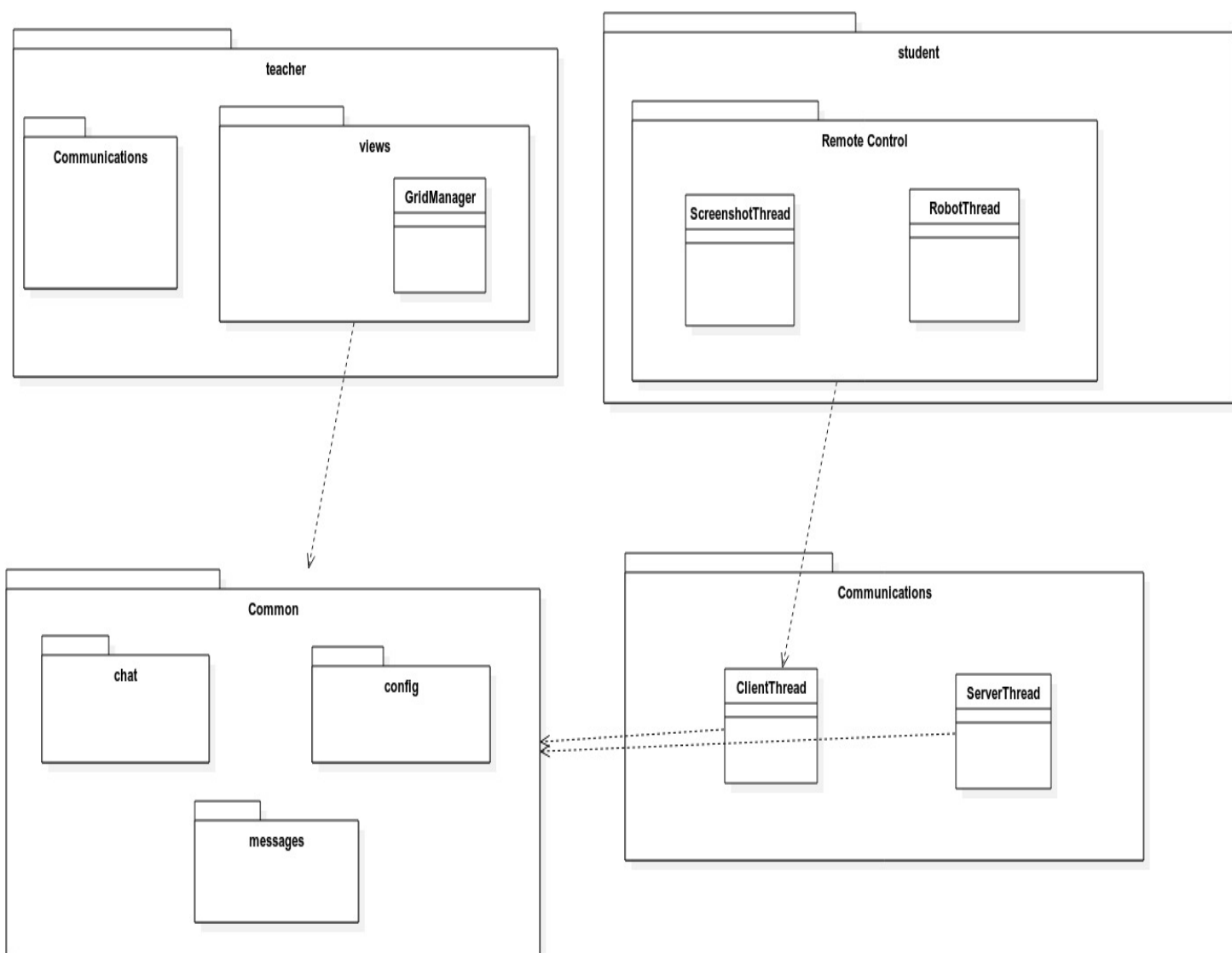
2 Diagramas



(Figura 1.1) Abstração da conexão entre computador do professor e dos alunos.



(Figura 1.2) Relação entre requisições e respostas de computadores em Contexto Estudante e em Contexto Professor



(Figura 1.3) Relação entre pacotes e classes do sistema.

3 Funcionamento planejado

Para realizar a instalação, configuração e uso do LabSpy em máquinas de um dado laboratório, os seguintes passos se fazem necessários:

1. **Instalação do LabSpy no computador do professor:** haverá um programa de *setup* para configurar o LabSpy no computador do professor. Durante o processo de instalação o administrador será orientado a criar um *hostname* ou um *IP fixo* para o computador em questão (o computador que rodará em Contexto Professor). Após finalizado, o processo fará com que o computador inicie junto com um “processo-servidor” do LabSpy, este processo permitirá que os computadores em Contexto Estudante se conectem, enquanto iniciam, ao computador do professor, estabelecendo assim uma comunicação entre os mesmos.
2. **Instalação do LabSpy no computador do aluno:** haverá um programa de *setup* para configurar o LabSpy nos computadores dos alunos. O *setup* deve ser feito em cada máquina onde é desejado o uso do LabSpy no Contexto Estudante. Esse processo de instalação fará com que o computador em questão sempre inicie com um “processo-cliente” do LabSpy que fará a conexão com o computador em Contexto Professor. Dessa forma, ao iniciar, o sistema em Contexto Estudante começará a comunicação com o computador em Contexto Professor.
3. **Aguardando conexões de computadores em Contexto Estudante:** após ser inicializado, o computador em Contexto Professor terá que aguardar pelo início dos computadores em Contexto Estudante, que ficarão responsáveis por iniciar a comunicação entre computadores *estudante-professor*. Contudo, na ocorrência de uma situação contrária (ou seja, o computador do estudante iniciar antes do computador do professor), o computador em Contexto Estudante terá que esperar até que o computador em Contexto Professor seja iniciado. A partir desse momento, o “servidor” (computador do professor) existirá e será possível que o “cliente” (computador do estudante) faça o início da comunicação. Um computador em Contexto Estudante sempre é responsável por iniciar a comunicação.
4. **Novas conexões de computadores em Contexto Cliente ao computador em Contexto Professor (ações em uma nova conexão):** para cada nova conexão iniciada pelo computador estudante, o computador em Contexto Professor vai criar uma thread cliente. Essas “threads clientes” serão utilizadas pelo computador em Contexto Professor para fazer requisições aos computadores em Contexto Estudante. Através delas será possível separar a parte “servidor” da parte “cliente” no computador em Contexto Professor.
5. **Requisições do professor:** quando uma requisição for feita pelo computador em Contexto Professor, ela será transmitida pela “thread cliente” respectiva ao computador estudante em questão, que tratará essa requisição e fornecerá a sua respectiva resposta ou ação.

4 Segunda entrega

4.1 Novidades

- Refatoramento do código, professor passa a ser cliente, e o aluno, servidor;
- Nova classe, `BaseClientThread`, responsável pela comunicação assíncrona entre aluno e professor;
- Instalação remota do programa via SSH;
- Possibilidade de controle remoto do computador do aluno através do computador do professor;
- Comunicação assíncrona entre aluno e professor, portanto, quando um aluno envia uma mensagem, ele não necessita ficar bloqueado esperando resposta do professor.

4.2 Pré-requisitos

Esta primeira versão do LabSpy possui alguns pré-requisitos que são necessários para que o sistema funcione corretamente.

- 1) Pré-requisitos gerais (necessário para ambos os computadores)
 - Oracle JRE 1.8_45, disponível no [site oficial](#) do Java;
 - Sistema operacional Ubuntu utilizando Upstart (apenas anteriores ao Ubuntu 15.04).
 - Caso deseje *compilar* o projeto em *.jar*, também é necessário JDK 1.8_45 e Ant 1.9.4.
- 2) Pré-requisitos específicos (computadores dos estudantes)
 - Um servidor SSH instalado, configurado e rodando (apenas para instalação);
 - Sistema não-*headless*, ou seja, com um servidor X instalado (Xorg Server, por exemplo).
- 3) Pré-requisitos específicos (computador do professor)
 - Acesso aos computadores dos estudantes como *root*;

4.3 Comunicação

Há dois processos em comunicação para manter a sincronização e o correto funcionamento do monitoramento remoto. A comunicação é feita entre os dois processos de forma assíncrona, isto é, assim que um envia uma requisição, este não necessita estar bloqueado esperando a resposta do outro processo. Neste projeto, a classe que é responsável pela conversação entre cliente (professor) e servidor (aluno) é a `BaseClientThread`, e a comunicação é feita através de sockets. Por cliente e servidor, entende-se que servidor é o processo que aguarda conexões provenientes de outros processos (ou seja, é passivo), enquanto que cliente é o processo que inicializa a conexão a um servidor (ou seja, é ativo). Destacamos isso pois, após o estabelecimento da comunicação, dados são tanto transmitidos, quanto recebidos, de ambos os lados da conexão. Para o envio das mensagens, buffers são usados como caixa postal para garantir o funcionamento correto de escritas e leituras entre os dois processos; uma requisição é enviada ao buffer, que é enviado ao destinatário, que pode então responder (escrevendo uma

requisição como resposta) ou apenas ler os dados em questão. Essas requisições tem tamanho variável, portanto, limites são traçados de forma que não hajam inconsistências e/ou inconveniências durante a troca de mensagens, que são enviados de maneira sequencial e possuem o tamanho especificado logo no início da requisição.

4.3.1. Buffers

Para o buffer de leitura de dados, um espaço fixo de 128 bytes é alocado, para indicar o tamanho do dado a ser lido, enquanto o segundo espaço, é reservado à mensagem em si.

Para o buffer de escrita, uma requisição é selecionada de uma queue de requisições, e depois, o socket escreve no buffer de escrita, sequencialmente.

4.3.2 Leitura e escrita

A leitura é realizada de forma com que, de princípio, haja uma verificação no tamanho do buffer de leitura (readSize), que, se possuir valor diferente de -1, indica que está no meio do processamento de uma mensagem com tamanho igual a readSize. Porém, caso o tamanho (readSize) possua valor diferente de -1, um tamanho é definido a ele de acordo com o início da próxima mensagem (pois uma mensagem está para ser recebida). Sabendo o tamanho da próxima mensagem, a mensagem é carregada em pedaços definidos automaticamente pelo sistema operacional e, quando totalmente carregada, a mensagem a ser lida é descompactada e então interpretada usando uma classe nativa do Java chamada ObjectInputStream, que desserializa a mensagem e passa para o método receiveMessage tratar.

A escrita de uma mensagem é feita com verificação se o buffer do sistema está disponível para escrita, se sim, uma requisição de uma queue de requisições é selecionado. Se não houver mais requisições de escrita, a verificação do buffer de escrita do sistema é desativada, senão, o buffer do sistema operacional é verificado tanto para escrita quanto para a leitura. Se a verificação do buffer de escrita está desativada e o método sendMessage é chamado, ele “acorda” a thread (usando uma operação wakeup) para que esta verifique o queue de requisições e então comece a verificar o buffer tanto para escrita quanto para leitura.

4.3.3. Envio de mensagens

Uma entrada necessita ser compactada antes de ser enviada ao pool de mensagens a enviar, para isso, o DeflaterOutputStream comprime os dados de entrada, enquanto o ObjectOutputStream escreve a mensagem. O buffer tem seu espaço alocado e, depois, a mensagem é enviada ao pool.

4.3.4 Recebimento de mensagens

O recebimento de mensagens varia, porque professor e aluno têm de receber mensagens específicas para realizarem as suas tarefas. Para o aluno (student/src/communication/ClientThread.java), ele pode receber instâncias de mensagens e operar de acordo com isto:

- Numa instância de *StartScreenshot*, é iniciada uma thread que envia screenshots de forma periódica, respeitando largura e altura especificados na operação;
- Numa instância de *ResizeScreenshot*, uma solicitação de redimensionamento do screenshot da tela do computador do aluno é feito, de forma que novas screenshots sejam enviados com novas largura e altura;
- Numa instância de *StopScreenshot*, o envio de screenshots é parado;
- Numa instância de *RobotMessage*, é enviada uma mensagem ao RobotThread, que pode ser relacionada a operações como mover o mouse, efetuar um clique ou pressionar uma tecla no teclado, por exemplo;
- Ou, se não for nenhuma das instâncias listadas, ele imprime a string contida na mensagem de forma que o usuário possa verificar o que causou ela.

O professor (teacher/src/communication/ClientThread.java) também receberá instâncias diferentes, atuando de uma forma específica:

- Numa instância de *Screenshot*, um screenshot é recebido e anexado como sendo o último screenshot do cliente em questão;

4.4 Visualizar Tela do Aluno

O professor poderá visualizar a tela do aluno por meio de uma comunicação assíncrona, descrita no tópico “Comunicação”. O aluno enviará mensagens para o professor consecutivamente por meio de sockets, o professor poderá ler essas mensagens e mostrá-las em sua tela, sempre atualizando de acordo com cada mensagem lida. Essa tela de visualização ficará em um grid com várias outras telas de outros computadores, se estes forem configurados, isto é, o professor poderá visualizar em uma única tela, todas as telas dos computadores configurados e poderá selecioná-las para uma visualização com uma dimensão maior.

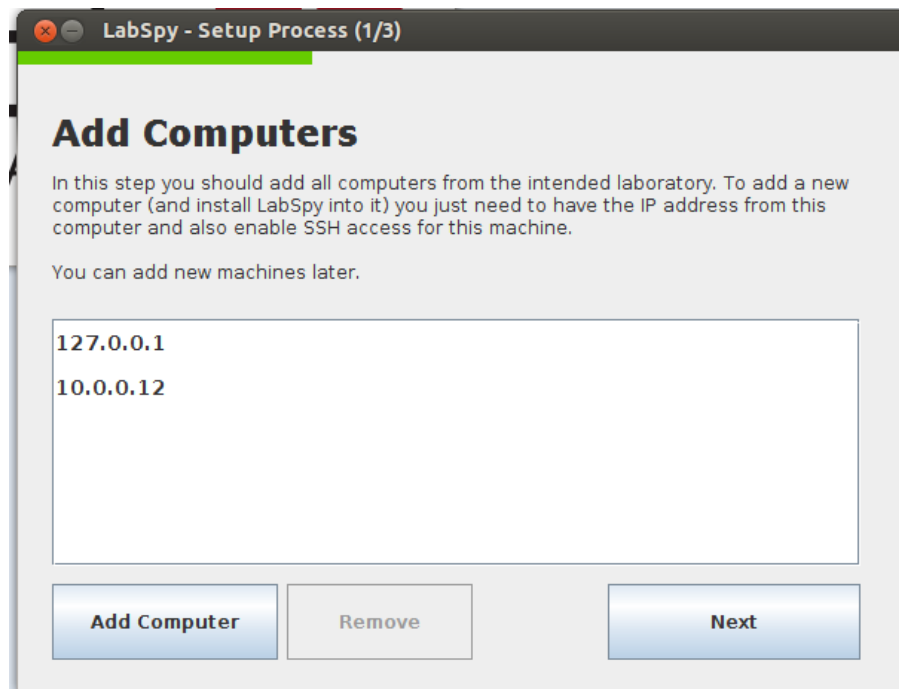
4.5 Controle Remoto

O professor poderá controlar remotamente o computador do aluno por meio de seu mouse e teclado. Ele terá uma grid com as imagens da tela dos alunos, como dito anteriormente, e cada tela será selecionável, possibilitando que seja selecionada apenas o computador que o professor queira controlar remotamente. Cada visualização selecionável é relacionada com o IP da máquina que está enviando screenshots, fazendo uma comunicação direta com a máquina que se quer controlar.

4.6 Processo para iniciar e usar as funcionalidades

O LabSpy em sua versão atual já fornece suporte para instalação remota via SSH. Para começar a instalação basta utilizar a interface gráfica do programa e clicar em “Functions -> Configuration”. Nota: caso seja a primeira vez que o programa esteja sendo executado, o LabSpy vai perguntar se o usuário deseja configurar os computadores. Ao clicar em “Configuration”, existem 3 etapas para configurar os computadores dos estudantes:

1ª Etapa: o usuário adiciona os endereços IP dos computadores desejados, ou seja, o endereço IP dos computadores dos estudantes.



The screenshot shows the 'LabSpy - Setup Process (1/3)' window. The title bar includes standard window controls and the text 'LabSpy - Setup Process (1/3)'. The main heading is 'Add Computers'. Below it, a paragraph explains that the user should add all computers from the intended laboratory and that adding a new computer requires its IP address and enabling SSH access. A sub-note states 'You can add new machines later.' Below this is a text area containing the IP addresses '127.0.0.1' and '10.0.0.12'. At the bottom, there are three buttons: 'Add Computer', 'Remove', and 'Next'.

Add Computers

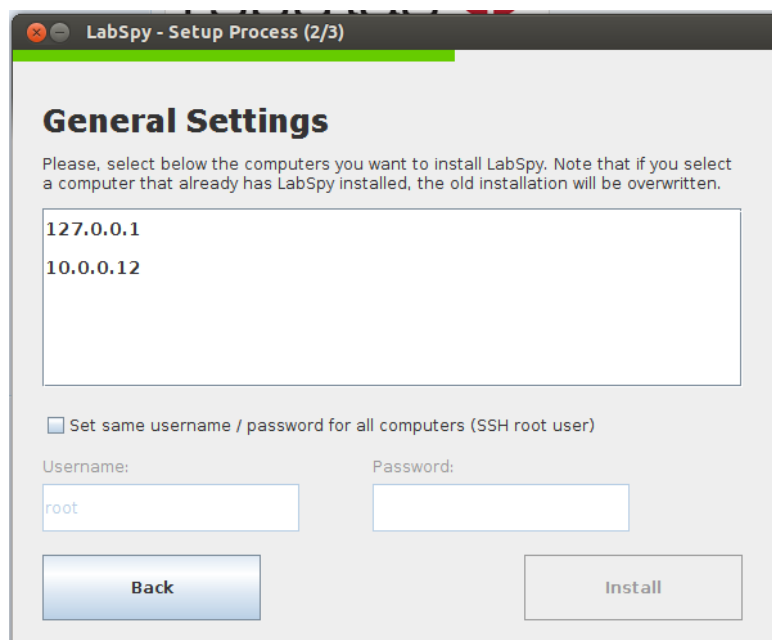
In this step you should add all computers from the intended laboratory. To add a new computer (and install LabSpy into it) you just need to have the IP address from this computer and also enable SSH access for this machine.

You can add new machines later.

127.0.0.1
10.0.0.12

Add Computer Remove Next

2ª Etapa: permite que o usuário selecione os computadores adicionados que terão o LabSpy adicionado. Além disso é possível definir um *username* e *senha* universal (para todos os computadores) para o processo de login via SSH. Caso os computadores dos estudantes tenham as mesmas credenciais para a conta de root, esta é uma opção que pode economizar muito tempo.



The screenshot shows the 'LabSpy - Setup Process (2/3)' window. The title bar includes standard window controls and the text 'LabSpy - Setup Process (2/3)'. The main heading is 'General Settings'. Below it, a paragraph instructs the user to select computers for installation and notes that existing installations will be overwritten. Below this is a text area containing the IP addresses '127.0.0.1' and '10.0.0.12'. A checkbox labeled 'Set same username / password for all computers (SSH root user)' is checked. Below the checkbox are two input fields: 'Username' (containing 'root') and 'Password' (empty). At the bottom, there are two buttons: 'Back' and 'Install'.

General Settings

Please, select below the computers you want to install LabSpy. Note that if you select a computer that already has LabSpy installed, the old installation will be overwritten.

127.0.0.1
10.0.0.12

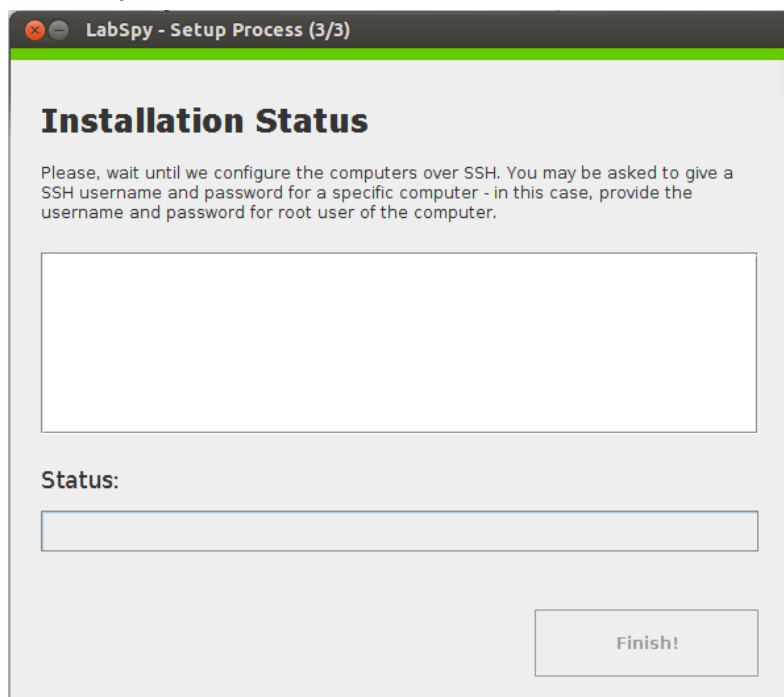
☒ Set same username / password for all computers (SSH root user)

Username: Password:

root

Back Install

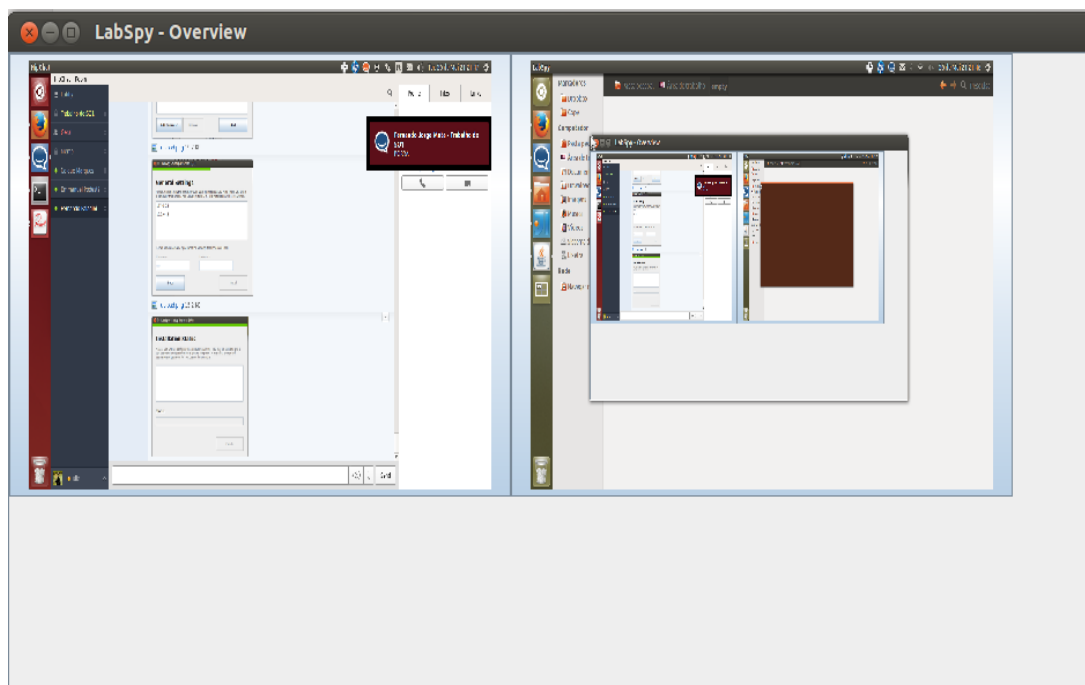
3ª Etapa: todos os computadores selecionados terão o LabSpy instalado e configurado. Caso a opção de *username* e *senha* universal não tenham sido selecionados, para cada computador o sistema pedirá informações de *username* e *senha*.

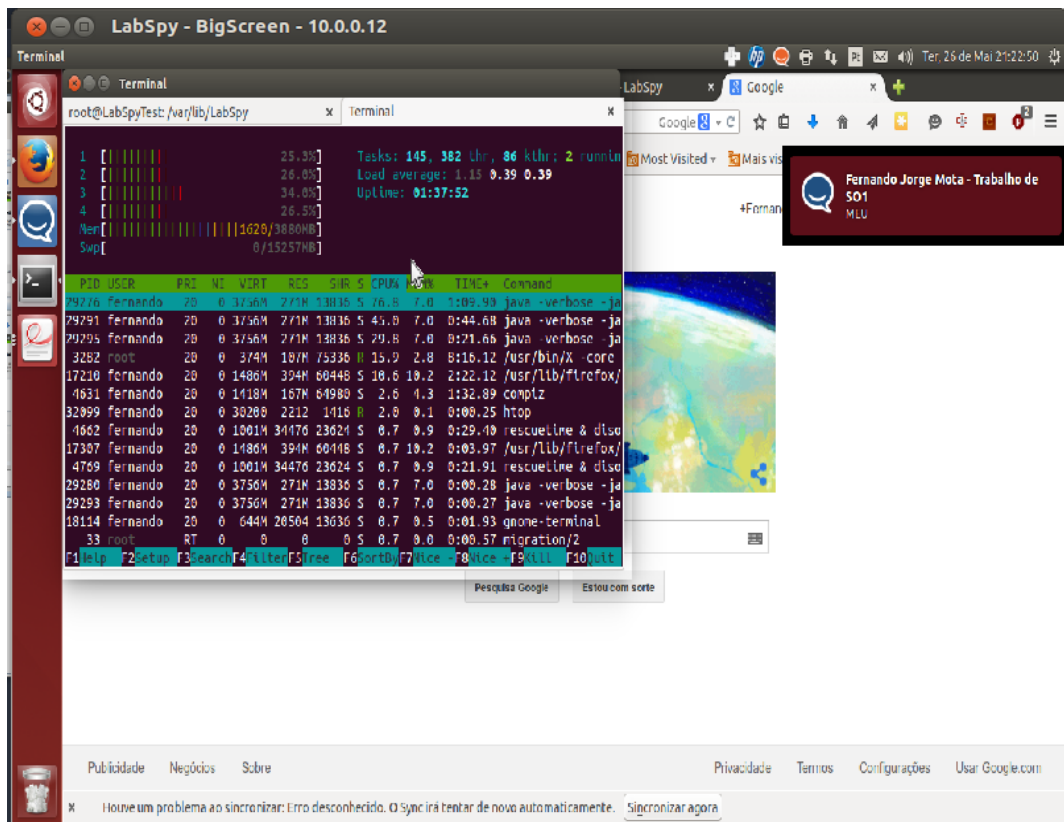


De forma mais técnica, a conexão e instalação via SSH consiste no envio dos arquivos “assets/labspy.sh” e “out/artifacts/Student/Student.jar” para a pasta “/var/lib/LabSpy/” no computador remoto. O primeiro arquivo é um script que permite realizar um comando de *start* e *stop* no Student.jar, que consiste em um servidor cuja o qual o computador do professor se conectará. Após a transferência dos dois arquivos para o computador remoto, um symlink é criado do arquivo /var/lib/LabSpy/labspy.h para /etc/init.d/labspy_client. Um último comando que utiliza o Upstart é feito para configurar o labspy_client como um script a ser iniciado junto com o sistema (em seu boot). A partir de agora, toda vez que o computador do estudante iniciar, o Student.jar (que é parte do LabSpy) iniciará junto com o sistema, permitindo assim que o computador do professor possa realizar uma conexão e acessá-lo remotamente.

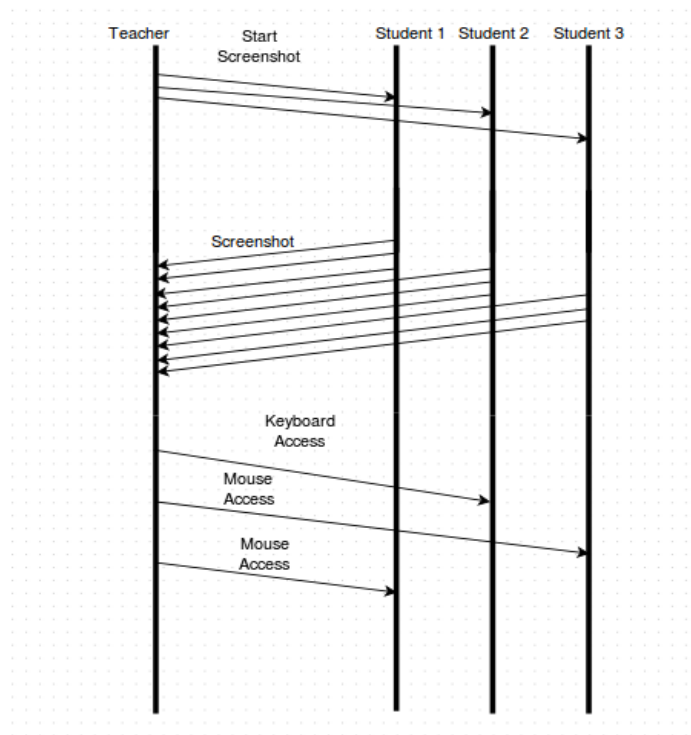
Após realizar a configuração das máquinas (incluindo reinício do sistema para que o Student.jar possa estar rodando) basta clicar em “Overview -> Screenshot” para começar a visualizar a tela dos computadores dos estudantes. Você verá uma nova tela com uma grid com a tela dos computadores configurados já enviando mensagens com suas screenshots, ou seja, mostrando o que está na suas telas para o computador do professor. Caso o docente queira controlar remotamente um computador do aluno, é possível selecionar a tela do computador que se quer controlar e será aberta uma nova janela com a tela do computador selecionado. Após esta ação, o professor já estará controlando o computador do aluno, podendo escrever com o teclado e mexer com o mouse no computador do aluno.

Imagens do programa executando:

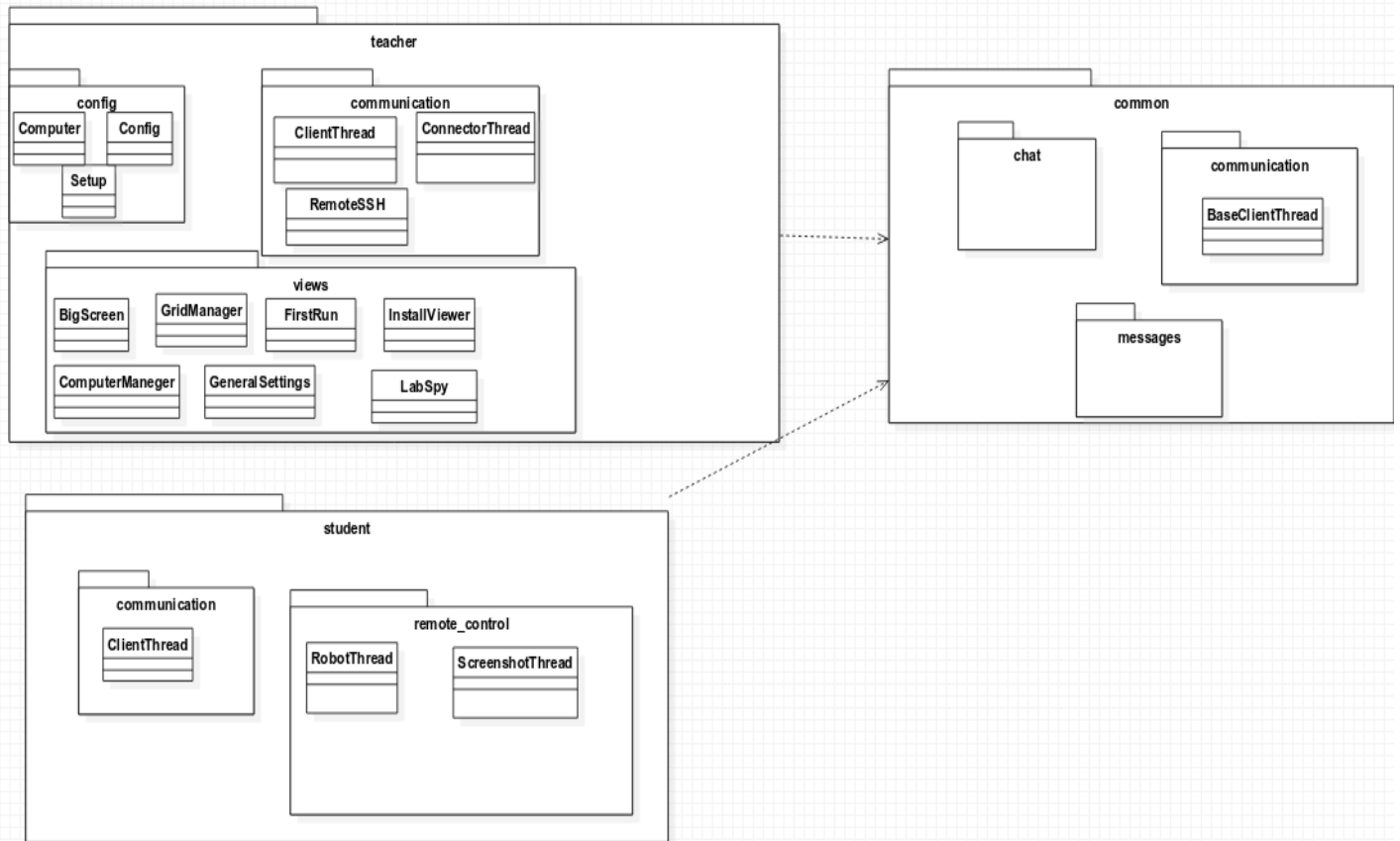




5 Diagramas



(Figura 1.1) Relação entre requisições e respostas de computadores em Contexto Estudante e em Contexto Professor



(Figura 1.2) Relação entre pacotes e classes do sistema.